VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Porovnání shlukovacích algoritmů v nástroji WEKA

Comparison of Clustering Algorithms in WEKA tool

2014 Vít Poledna

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Vít Poledna**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Porovnání shlukovacích algoritmů v nástroji WEKA
Comparion of Clustering Algorithms in WEKA tool

Zásady pro vypracování:

Nástroj WEKA je volně širitelný nástroj pro vizualizaci a analýzu dat. Shlukovací algoritmy jsou jedny z velmi častých algoritmů používaných v analýze dat.
Cílem práce je otestovat shlukovací algoritmynástroje WEKA na doméněsociálních sítí.

1. Prozkoumejte a popište nástroj WEKA.
2. Prozkoumejte a popište různé shlukovací algoritmy.
3. Vyhledejte datovou kolekci v oblasti sociálních sítí.
4. Porovnejte zatížení paměti, procesoru a výsledky po použití různých shlukovacích algoritmů v nástroji WEKA.
5. Popište výsledky shlukovací analýzy nad vybranou kolekcí.

Seznam doporučené odborné literatury:

Witten I.H., Frank E., Hall M.A.: Data Mining: Practical Machine Learning Tools and Techniques, 2011 by Morgan Kaufmann Publishers (ISBN: 978-0-12-374856-0)

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Petr Berek**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.*

V Ostravě 29.4.2014 .....

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29.4.2014 .....

Na tomto místě bych chtěl poděkovat Ing. Petru Berkovi za vedení této bakalářské práce. Dále také za pomoc, cenné rady, trpělivost a ochotu, kterou mi v průběhu zpracování bakalářské práce věnoval.

Abstrakt

Tato bakalářská práce se zabývá algoritmy pro shlukování dat. Jejich představením, otestováním na kolekci dat z oblasti sociálních sítí pomocí nástroje WEKA. Cílem této práce je porovnání těchto algoritmů v závislosti na využití zatížení procesoru, využití paměti a hodnoty výpočetního času potřebného k provedení shlukování. V této práci bylo zjištěno, že tyto tři sledované parametry se u jednotlivých sledovaných algoritmů výrazně liší.

Klíčová slova: Shlukování, WEKA, COBWEB, DBSCAN, EM, K-MEANS, FARTHEST FIRST, OPTICS, sociální síť

Abstract

This thesis deals with algorithms for clustering data. Their performance and testing for the collection of data from social networks using WEKA tool. The aim of this thesis is to compare these algorithms based on the use of CPU load, memory usage and value of computing time required for clustering. In this work it was found out that the three experimental parameters for each monitored algorithms are significantly different.

Keywords: Clustering, WEKA tool, COBWEB, DBSCAN, EM, K-MEANS, FARTHEST FIRST, OPTICS, social network

Seznam použitých zkratek a symbolů

WEKA	–	Waikato Environment for Knowledge Analysis
ARFF	–	Attribute-Relation File Format
SNAP	–	Stanford Network Analysis Project
DBSCAN	–	Density-based spatial clustering of applications with noise
EM	–	Expectation–Maximization
OPTICS	–	Ordering points to identify the clustering structure

Obsah

Seznam tabulek

Seznam obrázků

1 Úvod

Tato bakalářská práce pojednává o shlukování a algoritmech používaných k této činnosti. Shlukování je proces, během kterého probíhá porovnávání jednotlivých objektů a jejich přiřazení podle míry podobnosti do tzv. shluků. Cílem shlukovací analýzy je nalézt skupiny objektů, které se od sebe budou co možná nejvíce lišit a zároveň data obsažená v jednotlivých shlucích si budou navzájem co nejvíce podobná. Pro shlukovací analýzu existují různé metody shlukování. Můžeme je rozdělit na metody hierarchické a metody nehierarchické. Při použití hierarchických metod dochází k dalšímu využívání již nalazených shluků. Uplatnění tohoto druhu metod je možné nalézt v bioinformatice, dokumentografických informačních systémech či při zpracování obrazu. Typickým zástupem algoritmů patřících do této kategorie je algoritmus COBWEB. Naopak u nehierarchických metod dochází k vytvoření shluků v rámci jednoho kroku. Využíti nehierachických metod je například pro rozpoznávání řeči a rukopisu. Typickými zástupci jsou algoritmy K-MEANS, DBSCAN a OPTICS.

Pro shlukovací analýzu v rámci této práce byl vybrán nástroj WEKA. Tento nástroj vznikl na půdě univerzity Waikato na Novém Zélandu na počátku devatesátých let minulého století. Nabízí výběr nejznámějších shlukovacích algoritmů a využívá je pro shlukování. Kromě výpisu výsledku shlukovaní zobrazuje i grafickou reprezentaci dat. Je to sice velmi jednoduchý nástroj, nicméně v rámci jeho funkčnosti nabízí velké množství možností pro tzv. dolování dat z rozlehlých sítí.

Aby bylo možné vybrané algortimy otestovat, bylo nutné nalézt datovou kolekci, která by byla vhodná pro shlukování. V rámci této práce proběhne testování na datech z projektu Standfordské univerzity pro rozlehlé sítě, které jsou volně dostupné na stránkách projektu. Navíc se tato práce okrajově zabývá i představením sociálních sítí.
Na závěr práce je provedeno porovnání jednotlivých shlukovacích algoritmů.

Tato práce je rozdělena celkem na 7 kapitol. První kapitolou je úvod, kde je představena základní vize celé práce.

V rámci 2 kapitoly je popsán a představen nástroj WEKA, který je používán pro praktickou část této práce a jehož výstupy jsou nutné pro porovnání jednotlivých algoritmů.

Kapitola 3 se zabývá rozdělením shlukovacích metod a představením jednotlivých shlukovací algoritmů používaných v nástroji WEKA.

V kapitole 4 jsou představeny sociální sítě a obsahuje úvod do analýzy těchto sítí.

Kapitola 5 obsahuje popis datové kolekce používané v rámci této práce. Dále jsou v této kapitole popsány operace, které bylo nutné provést nad datovou kolekcí, aby bylo možné tato data zpracovávat v nástroji WEKA.

Na závěr této práce probíhá představení výsledk; shlukové analýzy pro jednotlivé algoritmy a porovnání těchto algoritmů. V rámci závěru tedy probíhá shrnutí výsledků této práce a popsání jejího přínosu.

Práci ukončuje seznam používané literatury a příloha obsahující výsledky shlukové analýzy v grafické formě pro jednotlivé algoritmy.

2 Nástroj WEKA

Nástroj WEKA je open sourceový program pro zpracování dat, který primárně slouží k dataminingu. Vznikl na půdě univerzity Waikato (The University of Waikato in New Zealand) v roce 1993.

Jeho největší výhodou je to, že je šířen pod licencí GNU, tudíž je k dispozici široké veřejnosti. Mezi jeho hlavní disciplíny patří získávání dat z relačních databází a jejich další zpracování. Mezi tyto operace můžeme zařadit shlukovací operace, regresní analýzu, korelaci a další.

Původně byl celý nástroj naimplementován v programovacím jazyce C. Nicméně byl postupem času přepracován do jazyka JAVA. Velkou výhodou tohoto nástroje je to, že jej lze považovat za multiplatformní. Nástroj WEKA se může navíc pyšnit jednoduchým, ale silným grafickým vizuálem, který uživateli nabízí přehledně všechny potřebné operace. Nástroj WEKA má i své vlastní API, takže je možné jej vkládat jako jakékoliv jiné knihovny do dalších aplikací. [1]

2.1 Instalace nástroje

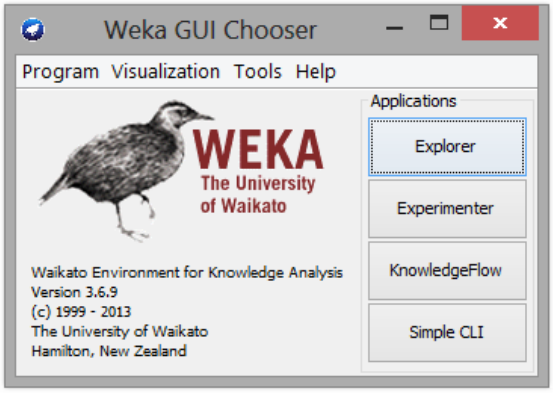
Nástroj WEKA lze stáhnout z webových stránek [1]. Kde lze nalézt ke stažení aktuální verze pro operační systémy Windows, Linux i MAC. Samozřejmostí je také výběr instalace pro 32bitové a nebo 64bitové verze těchto OS. Mimo to je možné si tento nástroj stáhnout i s JRE či bez. Každá sada instalačních souborů obsahuje i dokumentaci a uživatelský manuál.

2.2 Popis grafického rozhranní

Po dokončení instalace dochází k otevření nástroje WEKA v módu Chooser. V tomto kroku probíhá volba GUI rozhraní pro další práci.

- Simple CLI
  umožňuje spuštění nástroje bez použití grafického rozhraní a práci s programem pomocí okna terminálu
- Explorer
  spouští standartní grafické rozhraní
- Experimenter
  proti volbě Explorer navíc umožňuje provádění dalších experimentálních úkonů
- KnowledgeFlow
  na rozdíl od módu Explorer může provádět např. i úlohy bez načtení celého souboru do paměti

[1] http://www.cs.waikato.ac.nz/ml/weka/



Obrázek 1: Chooser mód nástroje WEKA

Při volbě módu explorer dochází k zobrazení základní obrazovky. Okno tohoto módu obsahuje záložky: [2]

- Preprocess
  Preprocessing slouží pro vybrání a definování dat, nad kterými má dále nástroj WEKA pracovat. Tyto datové soubory musí podléhat omezení na import. Nástroj WEKA dovoluje importovat soubory .arff, .csv, .names, .bsi a .data. Na tyto datové soubory lze dále spouštět filtry, kterými lze specifikovat data, se kterými má nástroj WEKA pracovat.

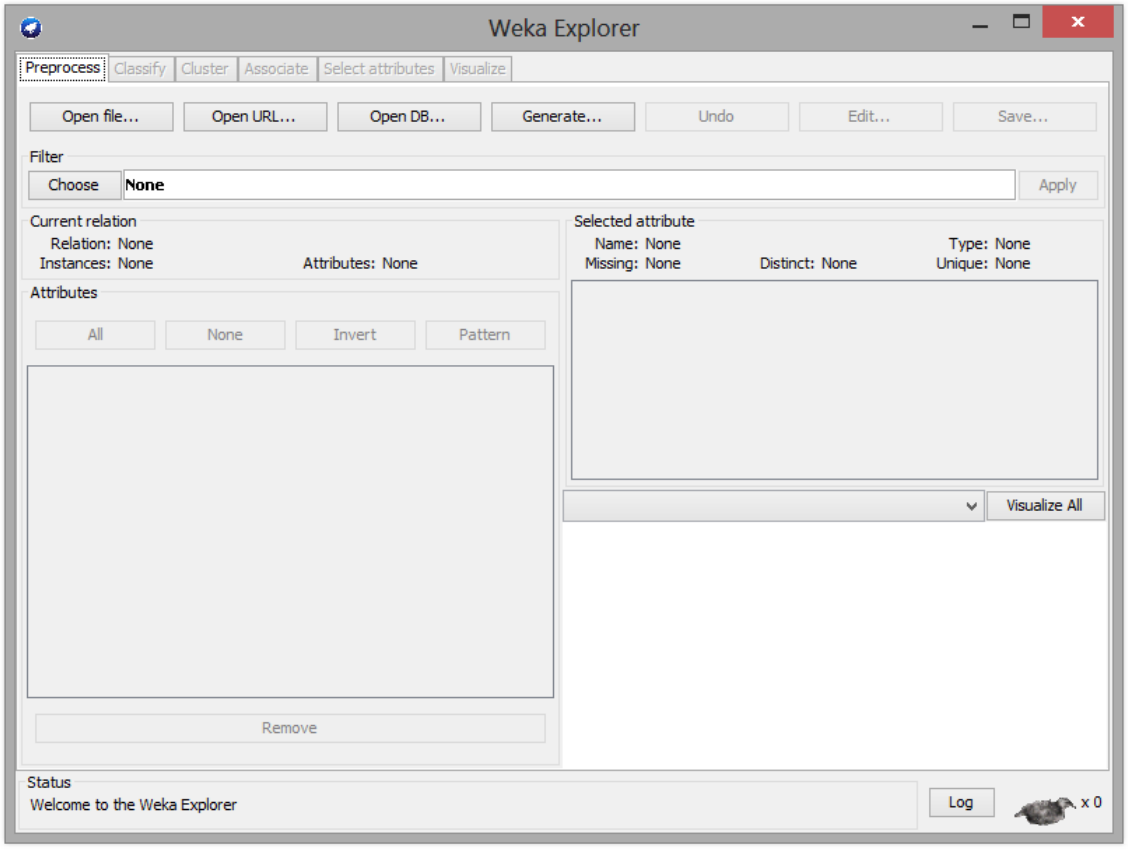
- Classify
  V sekci Classify dochází k použití sady předdefinovaných algoritmů pro interpretaci dat. Je totiž možné, že na úvod testování není možné vybrat správnou sadu algoritmů pro zpracování určených dat. Díky nástroji WEKA tedy lze určit, která posloupnost algoritmů se bude pro danou interpretaci nejlépe hodit.

- Cluster
  V části Cluster nástroj WEKA identifikuje společné znaky interpretovaných dat, díky kterým je snadnější přiřazení postupu dataminingu. Uživatel má v této části i možnost ignorování některých atributů výchozího souboru. Což může být vhodné při zpracovávání dat, která se v některých částech shodují a tyto části nejsou součástí určeného sledování.

- Association
  Slouží pro výběr možností sdružení ve vybraném datovém souboru.

- Select attributes
  Kategorie Select attributes se používá pro přímé specifikování atributů, které mají být konkrétně sledované či zpracované. Program nejprve vyhodnotí zpracovávána data dle vybraných atributů a poté hledá shodná data, díky kterým by mohl být výpočet zrychlen.

- Visualize
  Poslední kategorie slouží pro grafickou interpretaci dat analyzovaných v nástroji WEKA.



Obrázek 2: Explorer mód nástroje WEKA

2.3 Struktura ARFF souboru

Pro zpracování dat v nástroji WEKA se používají typy souborů .arff, .csv, .names a další, uvedené výše. Pro analýzu v rámci této práce byl vybrán jako nejvhodnější typ souboru právě formát ARFF.

Typ souboru ARFF (Attribute-Relation File Format) byl vyvinut na katedře informatiky univerzity Waikato. Primárně byl tento typ souboru vyvinut pro jeho snadné použití v nástroji WEKA.

ARFF je textový formát typu ASCII, který obsahuje výčet atributů a jejich instanci, které lze dále zpracovávat. Tento soubor je rozdělen na dvě různé časti.

První část je záhlaví souboru, které obsahuje název souboru a výčet atributů používaných ke zpracování. Tato sekce je označena *@relational nazev* a *@atribute atribut jmeno, datovy typ*. Datový typ atributu může být typu number, string, date a také může být nahrazen výčtem hodnot, kterých může daný atribut nabývat, označených složenými závorkami.

Druhá část ARFF souboru obsahuje jednotlivé instance pro výše definované atributy. Celá sekce začíná klíčovým slovem *@data*, pod kterým následují jednotlivé instance. Tyto instance jsou rozděleny do jednotlivých řádků, tedy každý řádek reprezentuje jednu instanci. V případě chybějících dat je možné tyto data nahradit symbolem ?. Dále také veškeré data typu string či data definovaná výpisem všech možných hodnot, jsou caresensitive. Speciálním znakem % můžeme také do ARFF souboru přidat libovolný komentář.

V rámci praktické části této práce bude probíhat testování na jednotlivých ARFF souborech, které patří do kolekce dat, získané ze SNAP (Stanford Network Analysis Project). Tuto kolekci bylo pro potřeby našeho testování upravit, což bude podrobněji popsáno v kapitole č. 5. Původně bylo vycházeno z několika souborů, obsahující separátní data - *fileName.edges, fileName.circle, fileName.feat, fileName.egoFeat a fileName.featNames*. Jednotlivé soubory budou taktéž podrobněji rozebrány v kapitole č. 5.

3 Shlukovací algoritmy používané v nástroji WEKA

Shluková analýza popisuje metody a algoritmy, s jejíchž pomocí je možné jednotlivé sady objektů přiřadit do tzv. shluků (clusterů). Cílem shlukové analýzy je vytvořit více shluků, které se budou od sebe navzájem co možná nejvíce lišit a navíc data obsažená v jednotlivých shlucích si budou co možná nejvíce podobná. V rámci shlukové analýzy je možné nalézt vazby mezi jednotlivými objekty, které nemusí být vysvětleny či nějak dále interpretovány. [3]

Metody shlukové analýzy můžeme rozdělit na:

- Hierarchické metody
  Při použití hierarchických metod dochází k dalšímu využívání již nalezených shluků k vytvoření shluků úplně nových. Průnikem dvou podmnožin základní množiny může být množina prázdná či množina existující v předešlém kroku. Hierarchické metody dále dělíme na metody aglomerační, rozkládací a konceptuální.
- Nehierarchické metody
  Pomocí nehierarchických metod dochází k vytváření shluků během jednoho kroku. Takto vytvořené shluky můžeme označit za disjunktní shluky.

3.1 Hierarchické shlukovací metody

Hierarchické shlukovací metody využívají dříve nalezených shluků pro vytvoření nových shluků. V rámci jejich činnosti vzniká jeden shluk, který obsahuje všechny objekty a navíc vznikají i skupiny shluků, kde každá z nich obsahuje jeden objekt. Graficky mohou být tyto činnosti vyjádřeny jako binární strom. Jednotlivé uzly stromu je možné označit za shluky. V rámci binárního stromu je pak možné jako vzdálenost mezi jednotlivými shluky označit svislý směr stromu, naopak vodorovný směr označuje rozklad shluků.

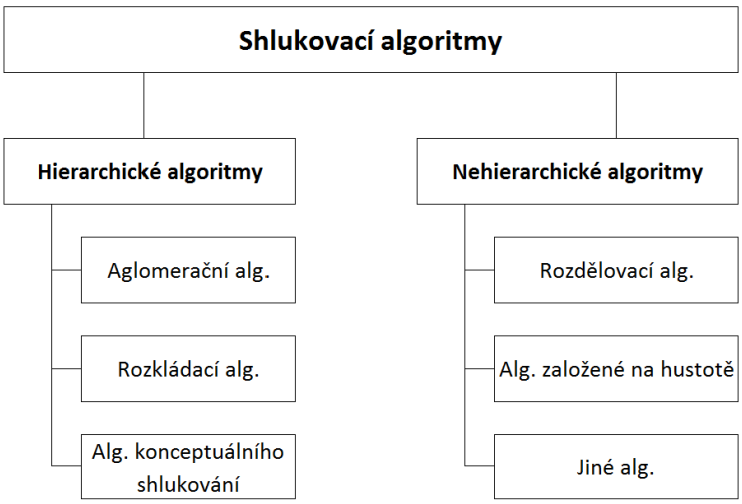
Hlavní nevýhoda metod tohoto druhu je to, že při své činnosti nevytvářejí jedinečné rozdělení datové kolekce, ale čistě hierarchii shluků. Navíc pro větší datové kolekce jsou považovány za velmi pomalé. Hierarchické metody tedy mohou být považovány za teoretické metody shlukovací analýzy, z jejíchž principů vznikla skupina metod založených na hustotě.

Hierarchické metody mají uplatnění ve zpracování záznamů síťové komunikace, v bioinformatice při analýze genů, v dokumentografických informačních systémech pro vyhledávání dokumentů např. podle tématu či při rozpoznávání vzorů během zpracování digitalizace obrazu. [4]

Mezi hierarchické algoritmy patří například algoritmy CLUSTER, CYRUS a COBWEB.

3.2 Nehierarchické shlukovací metody

Tyto metody vytváří všechny shluky během jednoho kroku, nevytvářejí hierarchickou strukturu. Při použití těchto metod dochází k rozložení analyzovaných dat do jednotlivých podmnožin, které probíhá s použitím předem definovaného atributu. Prvotní roz-



Obrázek 3: Rozdělení shlukovacích algoritmů

klad je dále upřesňován podle vzájemné vzdálenosti a odlišnosti jednotlivých shluků za předpokladu, že objekty mají být rovnoměrně rozložené do jednotlivých shluků. Většinou je tedy na úvodu analýzy definován rozklad, který se za pomocí těchto metod dále upřesňuje. Nehierarchické shlukovací jsou rozděleny podle počtu shluků na metody s pevně daným počtem shluků a s měnícím se počtem.

Pro určení optimálního počtu shluků se používá například C-index nebo Goodman-Kriskal index. Pro počáteční rozklad se zároveň používá postup, kdy probíhá rozklad náhodným přiřazením objektů do shluků. Poté probíhá přidružení jednotlivých objektů do všech shluků a vypočítání hodnoty kvality. Algoritmy používané v těchto metodách končí ve chvíli, kdy už nedochází ke zlepšení hodnoty kvality.

Druhým způsobem pro vytvoření počátečního rozkladu je vypočítání vzdálenosti k jednotlivým objektům k příslušným shlukům a jeho přiřazení k nejbližším shlukům. Takový postup končí, až ve chvíli kdy jsou po sobě jdoucí rozklady totožné.

Využití nehierarchických metod je možné nalézt například při rozpoznávání řeči, optických znaků či rukopisu. Dále i k analyzování chování návštěvníků jednotlivých webů či pro přiřazení dokumentů dle témat, zvlášť tehdy, pokud z velkého množství vstupních informací chceme získat relativně malý počet shluků.

Mezi nehierarchické algoritmy patří např. algoritmy K-MEANS, OPTICS, DBSCAN či FARTHEST FIRST.

Pro provedení shlukové analýzy v nástroji WEKA lze použít algoritmy uvedené v záložce cluster. Mezi ně patří COBWEB algoritmus, DBSCAN, EM a další.

3.3 COBWEB algoritmus

Algoritmus COBWEB vznikl z myšlenek dvojice Michalski, Stepp a byl zároveň inspirován algoritmy UNIMEM a CYRUS. Tento algoritmus používá hierarchické konceptuální shlukování. S jeho pomocí probíhá začleňování určených objektů do klasifikačního stromu.

Pro správu vyhledávání je v tomto algoritmu používaná heuristická míra, která je označovaná jako *kategorizační utilita*. [2] Tuto utilitu lze přirovnat k funkci, která ohodnocuje podobnosti či rozdíly jednotlivých objektů v různých třídách. Podobnost je definována podmíněnou pravděpodobností, tedy čím větší bude pravděpodobnost, tím větší část členů dané třídy bude sdílet určenou hodnotu. Naopak čím je vyšší nepodobnost, tím je menší počet objektů sdílejících tuto hodnotu v dané třídě. V algoritmu COBWEB tedy probíhá snaha o získání co možná nejmenší hodnoty kategorizační utility, která se používá pro tvorbu klasifikačního stromu.

Navíc existuje i rozšíření algoritmu COBWEB. Jedná se o algoritmus CLASSIT, který slouží pro zpracování numerických atributů. Rozdíl mezi těmito dvěma algoritmy je právě v použití rozdílné kategorizační utility. [5]

Operátory používané v algoritmu COBWEB

Při provádění algoritmu COBWEB se v každém uzlu určuje klasifikace objektu, který stromem sestupuje. Na každé úrovni tento algoritmus umožňuje použití těchto čtyř operátorů:

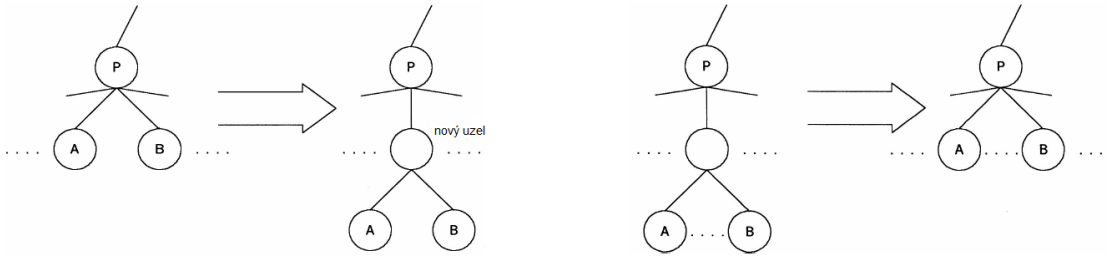
Vložení objektu do již existující třídy
Při použití algoritmu COBWEB je asi nejjednodušší možnost zkoumaný předmět vložit do již existují třídy. Hledání té nejvhodnější třídy probíhá pomocí již zmíněné kategorizační utility, díky které je stanoven nejvýhodnější uzel.

Vytvoření třídy nové
Při hledání nejvhodnějšího uzlu může nastat situace, kdy žádný z již vytvořených uzlů neodpovídá zkoumanému předmětu, proto může být vytvořena i třída nová. COBWEB v takovém případě musí rozhodnout, zda je kvalita určení nejvýhodnějšího uzlu dostatečná, nebo zda-li bude lepší vytvořit třídu novou.

Spojení dvou tříd do jediné společné třídy
Spojení dvou tříd se provádí za předpokladu, že nově vzniklý uzel bude dosahovat vyšší kvality, než dva původní uzly. V takovém případě dochází ke smazání spojovacího uzlu a jeho potomci jsou povýšeni o jednu úroveň.

Rozdělení jedné třídy na několik dalších tříd
Operátor rozdělení pracuje s předpokladem, že po provedení této operace dojde ke zvýšení kvality. V takovém případě probíhá vytvoření nového uzlu a sečtení hodnot uzlů, které se spojují. Takovéto dva uzly jsou dále nově připojeny k nově vytvořenému uzlu jako potomci.

[2] Odvození vzorce pro výpočet kategorizační utility [5]



Obrázek 4: Operátor sloučení, operátor rozdělení [5]

Výhody algoritmu

- je schopen provádět oboustranné hledání
- nová třída může být na rozdíl od použití algoritmu K-MEANS vytvořena za běhu

Nevýhody algoritmu

- předpokládá, že atributy jsou nezávislé jeden na druhém, což nemusí být pravda
- není vhodný pro zpracování většího množství dat

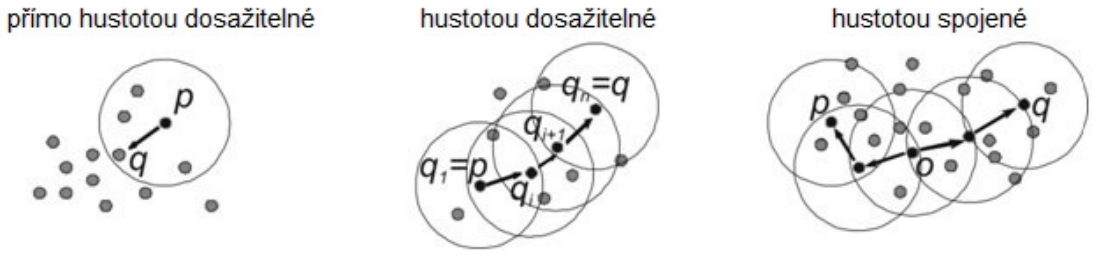
3.4 DBSCAN algoritmus

DBSCAN je shlukovací algoritmus založený na hustotě, který byl vyvinut v roce 1996. Zároveň patří mezi nejrozšířenější shlukovací algoritmy. DBSCAN pracuje na principu zjišťování počtu shluků od předpokládaného rozložení hustoty odpovídajících uzlů. Pracuje tedy tak, že určený bod stanoví jako součást clusteru dle jeho hustoty. Dále pak provádí zkoumání okolí tohoto bodu dle parametru ϵ. Body v jeho okolí jsou přiřazovány do kategorií dle jejich vlastní hustoty, popřípadě jsou přiřazeny do předešlé kategorie hustoty, pokud se tato hodnota rovná. Tento proces pokračuje, dokud nejsou všechny určené body součástí některé kategorie či dokud nejsou zbylé body vyhodnoceny jako šum. Pro svoji činnost pracuje s dvěma základními parametry. [7]

- ϵ - maximální rádius okolí bodu
- MinPts - minimální počet prvků v okolí

Objekty můžeme definovat dle možnosti dosažitelnosti na:

- Přímo hustotou dosažitelné (Directly density-reachable)
  Objekty na hranici jsou přímo hustotou dosažitelné z jádra objektu. Naopak to ale neplatí. Pouze v případě, že budou oba objekty jádra, bude jeden přímo hustotou dostupný z druhého a opačně.



Obrázek 5: Možnosti dosažitelnosti [7]

- Hustotou dosažitelné (Density-reachable)
  Objekty na hranici jsou hustotou dosažitelné z jádra objektu. Naopak to neplatí. V případě, že budou oba objekty jádra, jeden bude přímo hustotou dosažitelný z druhého a opačně. Pro dva objekty nacházející se na hranici stejného clusteru totiž nemusí platit, že by jeden z nich byl hustotou dosažitelný s druhým objektem a opačně.
- Hustotou spojený (Density-connected)
  Dva objekty jsou hustotou spojené, pokud existuje nějaký objekt takový, že oba tyto objekty budou dosažitelné.

Výhody algoritmu

- nevyžaduje předem známý počet shluků
- vyžaduje pouze dva parametry ϵ a MinPts
- nalezne libovolně tvarované shluky
- nalezne shluky, které navíc mohou být obklopeny jiným shlukem
- nevyžaduje uspořádaná data v databázi, se kterou pracuje
- počítá i s možným šumem

Nevýhody algoritmu

- algoritmus založený na Euklidovské míře, která může být ve větších vzdálenostech nepoužitělná
- nepřesně zpracovává data, která obsahují velké rozdíly v hustotě

3.5 EM algoritmus

EM algoritmus byl definován v roce 1977. Tento algoritmus provádí iteraci dvou základních kroků Estimate a Maximize, z čehož je odvozen i jeho název. V rámci Estimate probíhá odhad hodnot nepozorovaných dat. Maximize naopak hledá maximální věrohodnost vzhledem k datům přes uvažované modely. Při každém kroku dochází ze zvýšení věrohodnosti získaného modelu. [8]

Nalezení takovéhoto maximálního věrohodnostního ohodnocení vyžaduje přijetí derivace věrohodnostní funkce s ohledem na veškeré neznámé hodnoty. Právě schopnost vypořádat se s chybějícími daty či sledování neznámých proměnných patří mezi největší výhody tohoto algoritmu. Používá se například v případech, kdy známe model, ale některé veličiny nejsme schopni pozorovat. Často tento algoritmus využívá směsi Gausovských rozložení např. na popis funkce při zpracování obrazu.

Speciální případem toho algoritmu je algoritmus K-MEANS.

Výhody algoritmu

- hodí se na použití při shlukování menších datových kolekcí a to i v případě chybějících dat

Nevýhody algoritmu

- průběh algoritmu je velmi složitý

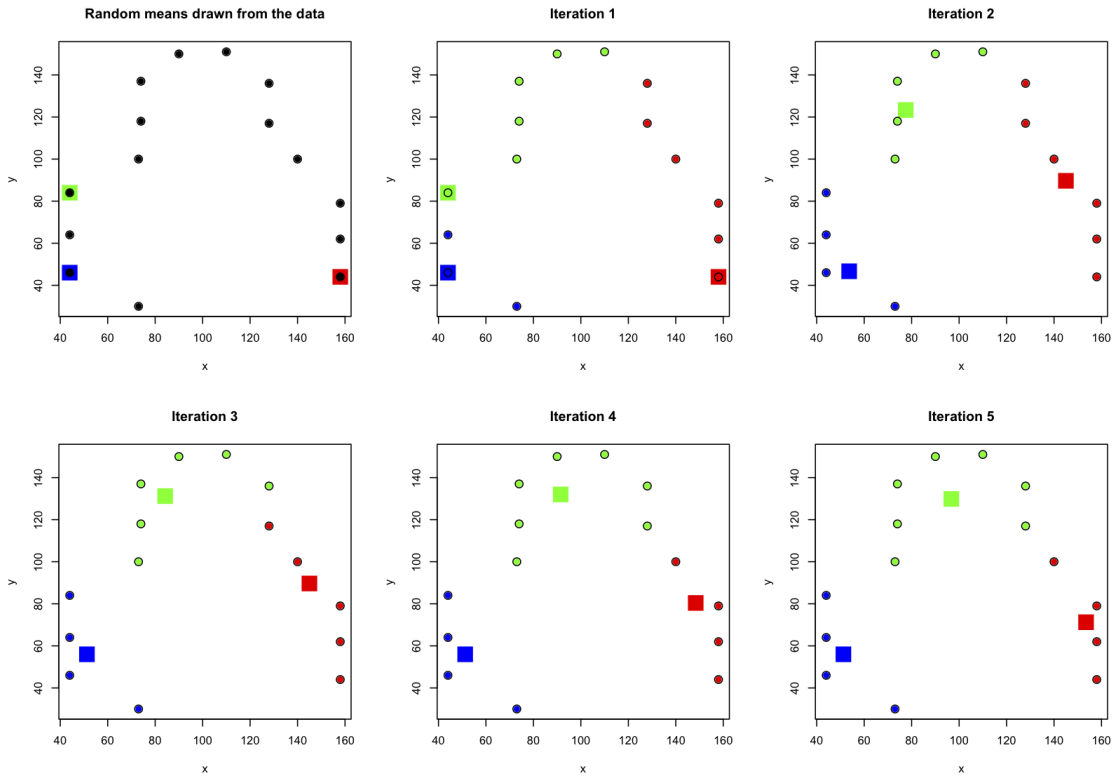
3.6 K-MEANS algoritmus

Algoritmus K-MEANS je jeden z velmi jednoduchých algoritmů pro shlukování, který se často používá. Je navíc známý i pod jinými názvy - MacQueenova metoda či Llodův algoritmus.

K-MEANS algoritmus začal být používán v padesátých letech 20. století. Patří mezi algoritmy dělící metody. V rámci jeho činnosti probíhá převod množiny dat na fixní definovaný počet k shluků. [9]

Celý postup tohoto algoritmu je založen na hledání tzv. k-středů. Nejprve jsou náhodným způsobem vybírány středy shluků, které se sobě co nejméně shodují. Je tedy možné zvolit různé středy shluků, díky čemuž je možné nad stejnou kolekcí dat získat i různé výsledky. Dále probíhá přiřazení takových objektů do sad objektů se stejným středem shluků. Později probíhá určení nových středu shluků a opětovné znovuzařazení objektů do jednotlivých shluků podle toho, jak moc se nepodobají novým středům shluků. Takovýto postup probíhá až do zjištění, že se středy již dále nemění.

Z důvodu větší efektivity bývá často určen parametr, do kterého je uložen maximální počet iterací. Pokud je takovéhoto počtu dosaženo, pak algoritmus končí. Algoritmus K-MEANS i další algoritmy pro svoji činnost potřebuji určit podobnost či nepodobnost dvou objektů. Pokud vezmeme v úvahu míru podobnosti, musí platit, že čím jsou si dva



Obrázek 6: Ukázka postupu shlukování algoritmem K-MEANS [10]

sledované objekty podobnější, tím je míra podobnosti větší. Právě z toho důvodu je definovaná tzv. Kosinova míra. Pro výpočet míry nepodobnosti používáme míry Euklidovskou a Manhattenskou.[3] [11]

U tohoto algoritmu je navíc možné specifikovat další používané varianty. Mezi ty nejpoužívanější patří:

- K-MEANS s opakovaným půlením
  Jde o velmi jednoduchou variantu. Probíhá dělení dané množiny na dvě části. Na úvod probíhá dělení celého vstupního souboru. Po té probíhá rozdělení jedné ze vzniklých množin. Ta je obvykle vybírána podle své velikosti - tedy podle největšího počtu prvků. Celý tento proces se opakuje, dokud algoritmus nedospěje k určenému počtu shluků.

- K-MEANS s prohazováním těžišť
  Každý shluk má definováno své těžiště a navíc je u každého shluku udržována množina kandidátů na těžiště v dalších krocích. Takovýmto způsobem se lze vyvarovat ukončení shlukování s lokálně nejlepším výsledkem.

[3] Odvození vzorců pro výpočet Kosinové, Euklidovské a Manhattenské míry [6]

K-MEANS algoritmus se často používá při výběru malého počtu shluků z velkého počtu objektů.

Výhody algoritmu

- za předpokladu, že jsou *k*-středy udržovány malé, je výpočetně rychlejší pro větší počet proměnných než hierarchické shlukování

Nevýhody algoritmu

- těžká předvídatelnost *k*-středů
- v případě použití nad shluky různých velikostí a různé hustoty není zcela přesný

3.7 FARTHEST FIRST algoritmus

FARTHEST FIRST je algoritmus rozdělovací metody. Jedná se o variaci algoritmu K-MEANS. Byl definován v roce 1985. Patří mezi dělící shlukovací algoritmy.

Tento algoritmus vybírá *k* objektů. Tyto objekty vybírá jako středy clusterů a následně přiřazuje ostatní objekty do příslušných clusterů. První střed clusteru je vždy vybírán náhodným způsobem. Naopak druhý se už vybírá tak, aby byl co nejméně podobný tomu předchozímu. Pro takovéto určení se používají výpočty tzv. mír podobnosti či nepodobnosti, které byly uvedeny již u předchozího algoritmu. [12]

Obecně lze tento algoritmus zařadit mezi nejlepší pro řešení K-centrum problému.

Výhody algoritmu

- výhodou toho algoritmu je jeho časová složitost, které je rovna θ (Nk) kde N je počet shlukovaných objektů a k je počet generovaných clusterů.

Nevýhody algoritmu

- nevýhodou je problém s určením počtu shluků, které má tento algoritmus vygenerovat

3.8 OPTICS algoritmus

Tento algoritmus patří mezi algoritmy založené na hustotě. Je velice podobný algoritmu DBSCAN. Na rozdíl od něj ale řeší jeho zásadní nedostatek. Tím je problém detekce smysluplných clusterů ve sledovaných datech s proměnou hustotou. [1]

Stejně jako výše zmíněný algoritmus DBSCAN pro svoji práci používá dva parametry - MinPts a ϵ. Dalším rozdílem je to, že algoritmus OPTICS nepřiřazuje sledovaným objektům členství v clusteru. Na rozdíl od toho ukládá pořadí, ve kterém jsou tyto údaje zpracovávány. Tento algoritmus pracuje i s dalšími dvěma údaji core-distance a reachability-distance.

Výhody algoritmu

- nalezne skupiny clusterů s nepravidelným tvarem i v oblastech s proměnou hustotou

4 Sociální sítě a jejich analýza

4.1 Sociální síť

Historie myšlenky sociálních sítí započala již v 19. století. Pánové E. Durbheim a F. Tonnies zjistili, že mezi jednotlivými aktéry v rámci sledované skupiny, dochází k vzájemným interakcím v oblastech společných zájmů, víry, osobních názorů atd.

Sociální síť je tvořena danými aktéry (fyzické osoby či organizace) a vazbami vzniklými jejich vzájemnými interakcemi. V rámci těchto vazeb pak lze objevit vzájemné vazby mezi například skupinami uživatelů, díky čemuž je možné objevit vzory podobnosti v celé struktuře takovéto sítě.

Za předchůdce sociálních sítí, jak jsou známy dnes, lze považovat síť Usenet, Arpanet aj. Tyto služby uživatelům umožňovaly zasílání krátkých zpráv dalším uživatelům či možnost zveřejnit svoji velmi jednoduchou webovou prezentaci.

Velký rozmach sociálních sítí započal na začátku nového tisíciletí, kdy vznikly např. sociální sítě MySpace, Badoo, Facebook, Twitter a další.

Dělení sociálních sítí

- Obecné/specializované
  Mezi obecné patří Facebook či Twitter, kde je možné sledovat další uživatele, jejich články, sdílet fotografie atd.
  Do kategorie specializovaných patří např. Last.fm – sociální síť obsahující internetové rádio, hudební encyklopedii, či systém pro doporučování hudby

- Otevřené/uzavřené
  Pro přístup do otevřených sítí je nutná pouze registrace, i když ani toto nemusí být podmínkou.
  Uzavřené sítě pro přístup vyžadují pozvánku od uživatelů dané sítě.

Sociální sítě mohou být děleny i dle způsobu použití:

1. osobní – Facebook, Twitter - osobní použití
2. profesní – Linked.in - pro budování profesního profilu např. při hledání práce
3. speciální – např. pro sdílení fotografií – Rajče.net

Společné znaky sociálních sítí

I když lze sociální sítě rozdělit do několika kategorií, všechny obsahují stejné základní znaky. [13]

Za první společný znak sociálních sítí je možné považovat sdružování lidí v kybernetickém prostoru. Tedy vytváření komunit uživatelů pro podporu komunikace, zábavy aj. Dalším společným znakem je i tzv. profil. Tedy prezentace daného uživatele obsahující jeho základní údaje, kontakt či další informace o něm. Nicméně pravděpodobně

nejdůležitějším společným znakem sociálních sítí je komunikace mezi uživateli. Ať už se jedná o komunikaci soukromou či o tu, která je zveřejněná i dalším uživatelům těchto sítí.

Na závěr představení sociálních sítí je ale nutné upozornit i na rizika používání těchto sítí. Kromě zneužití osobních údajů, fotografií či jiného obsahu je velkým problémem těchto sítí i anonymita. Žádným způsobem totiž u většiny sociálních sítí není kontrolováno, zda jsou uživatelé ve skutečnosti těmi, za které se pomocí svého profilu vydávají. Kromě této skutečnosti často dochází na těchto sítích i ke šíření poplašných zpráv - tzv. HOAXů.
Navíc bylo dokonce i pozorováno, že u některých lidí je možné z důvodu častého používání sociálních sítí, vytvoření závislosti na této činnosti.

4.2 Sociální síť Twitter

Sociální síť Twitter byla vytvořena v roce 2006 skupinou autorů (J. Dorsey, E. Williams, B. Stone, W. Glass). Ihned po svém spuštění zaznamenala prudký nárůst uživatelů. Ti mohou v rámci této sociální sítě sdílet tzv. tweety. Jedná se o krátké textové řetězce (délky maximálně 140 znaků).

Twitter je ovšem kromě sdělování krátkých statusů hlavně i sítí pro získávání informací. Ať už se jedná o sledování tweetů ostatních uživatelů, rodinných příslušníků či celebrit. V rámci této sítě lze mimo jiné sledovat i zpravodajství aj. Při sledování např. twitterové stránky serveru Novinky.cz tedy uživatel získává stručný přehled o aktuálním dění. Tuto funkčnost lze přirovnat k vyvolávání titulků článků od prodejce novin. Cílem takovýchto tweetů je tedy informovat a v případě zájmu uživatele mu nabídnout odkaz na stránku s podrobnějšími informacemi.

Sociální síť Twitter je možné tedy přirovnat například k programu ICQ, který byl rozšířen během posledního desetiletí. K této základní funkčnosti posílání krátkých zpráv Twitter obsahuje i další funkce popsané výše.

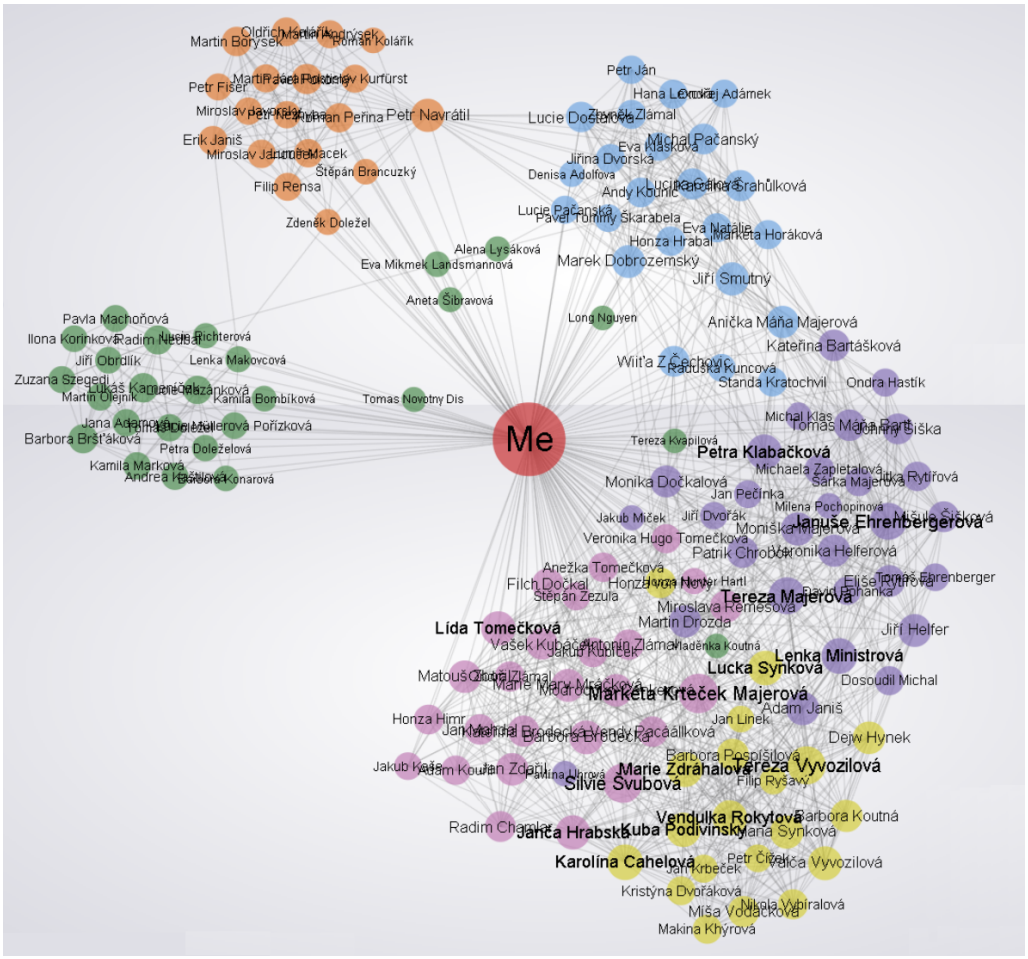
Na rozdíl od Facebooku je tedy Twitter podstatně jednodušší platformou, na které je možné si vytvořit svůj vlastní proud informací dle přání. Ve světě je tato sociální síť velice rozšířena, nicméně v České republice má nad ní podstatnou převahu Facebook. Počet uživatelů Facebooku v ČR se odhaduje na 3 milióny uživatelů, naopak u Twitteru se jedná o desítky tisíc uživatelů.

4.3 Analýza sociálních sítí

Analýza sociálních sítí, jak je známa dnes, započala na přelomu 70. a 80. let minulého století. K této skutečnosti přispěl velký rozmach počítačů v té době.

V rámci analýzy sociálních sítí se lze zaměřit na analýzu pomocí grafů či na analýzu statickými metodami. [14]

Při použití grafů bývá často kladen větší důraz na smysl a druh vazeb mezi jednotlivými aktéry než na samotného jedince či povahu jednotlivých druhů vazeb. Při použití



Obrázek 7: Ukázka grafu - vazby mezi přáteli na mém facebookovém profilu

grafů jsou tedy zobrazeny skupiny aktérů, kteří jsou spojeni vazbami a tvoří hustší shluky podle toho, kolik vazeb a aktérů takovýto prvek obsahuje.

Takovéto grafy dávají vizuální přehled o struktuře sociální sítě. Jedná se například o graf spolupráce, ve kterém jsou vyznačeny jak pozitivní, tak i negativní vazby mezi jednotlivými aktéry. Dále je také možné použít podepsané grafy. Díky nim lze předpovídat budoucí evoluci těchto grafu. Analýza pomocí statistických operací na druhou stranu používá vytvoření matice, ve které kromě aktérů záleží i na povaze vazeb.

Mezi hlavní vlastnosti, které jsou obvykle sledovány během analýzy jsou velikost, složení, struktura či obsah dané sítě.

V rámci analýzy je možné definovat pojem hustota sociální sítě. Jedná se o podíl uskutečněných vazeb ke vše možným vazbám mezi jednotlivými jedinci.

Analýzou sociálních sítí lze navíc i získat informace, které mohou být dále využity. Praktické využití takto získaných informací ze sociálních sítí je v současné době velmi rozšířeno. Může se jednat například o analýzu chování uživatelů pro marketingovou podporu či dokonce ve výzvědných službách různých zemí.

4.4 Sociální kruhy

V rámci sociálních sítí mohou uživatelé sami definovat skupiny aktérů (Facebook, Twitter – listy, Google+ - kruhy). Pro tyto skupiny mohou uživatelé určovat viditelnost svých jednotlivých příspěvků, fotografií aj. V obecném měřítku touto činností uživatelů vznikají skupiny přátel, známých atd. Ať se už jedná o skupinu rodinných příslušníků s těsnějšími vazbami mezi sebou či o skupinu lidí, kteří se ani neznají, nicméně mají nějakého společného aktéra či nějakou jinou činnost, vlastnost aj. Lze tedy získat skupiny uživatelů, které se nějakým způsobem prolínají. [14]

V současné době jsou uživatelé sociálních sítí přehlceni informacemi. Díky tomuto rozdělení svých známých na jednotlivé kategorie dochází ke zjednodušení při užívání sociálních sítí. Takovéto sociální kruhy jsou velmi specifické, právě z tohoto důvodu, že si je každý uživatel vytváří sám. Jsou tedy nezávislé na všech ostatních. Z tohoto důvodu můžeme pro analýzu takových kruhů použít shlukovací metody.

5 Popis datové kolekce v oblasti sociálních sítí a porovnání zvolených algoritmů

5.1 Popis datové kolekce v oblasti sociálních sítí

5.1.1 SNAP - Stanford Large Network Dataset

Projekt SNAP vznikl v roce 2004 jako výsledek výzkumné činnosti při zpracování dat z rozlehlých informačních a sociálních sítí. V této práci se zaměříme pouze na analýzu sociálních sítí.

Projekt SNAP zveřejnil na svých oficiálních stránkách [4] datové kolekce, které byly získané právě pro výzkumné účely. Z tohoto důvodu byla data anonymizována. Pro testování účinnosti vybraných shlukovacích algoritmů byla vybrána kolekce obsahující data ze sociální sítě Twitter.

Pro praktickou část této práce byla zvolena kolekce dat z výše zmíněného projektu SNAP pro sociální síť Twitter. Tyto data obsahují soubory:

- Edges
  soubor Edges obsahuje výpis jednotlivých vazeb mezi uzly
- Circles
  obsahuje sekvenci kruhů pro každý uzel, každý řádek obsahuje jeden kruh s id obsažených uzlů
- Feat
  v souboru Feat se nachází atributy obsažené v jednotlivých uzlech obsažených v souborech Edges
- EgoFeat
  soubor EgoFeat obsahuje sekvenci 1 a 0 pro jednotlivé FeatNames
- FeatNames
  tento soubor obsahuje výpis všech atributů používaných nad danou sítí

Sekvencí 0 a 1 se myslí vyjádření, zda je daný atribut obsažen (1) či zda není obsažen (0). Pro praktickou část této práce byly vybrány pouze soubory FeatNames a EgoFeat, jelikož obsahují nezpracované data vhodné pro shlukování různými algoritmy.

5.1.2 Postup úprav nad danou kolekcí

Aby bylo možné s daty vůbec pracovat, bylo nutné je nejprve převést do podoby, která by umožnila je analyzovat v nástroji Weka.

Veškeré soubory tedy bylo nutné upravit do typu souboru ARFF, který byl představen v úvodu této práce. [15]

[4] http://snap.stanford.edu

```
@relation pokus

@attribute  {0, 1}
@attribute ... {0, 1}
@attribute 1 {0, 1}
@attribute 1. {0, 1}
@attribute 11. {0, 1}
@attribute 2 {0, 1}
@attribute 2011 {0, 1}
.
.
.
.
.
.

@attribute zacwest {0, 1}
@attribute zadr {0, 1}
@attribute zeeg {0, 1}
@attribute zeldman {0, 1}
@attribute zephoria {0, 1}
@attribute zite {0, 1}
@attribute zsims {0, 1}

@data
0, 0, 1, 0, 0, 0, 0 ..... 0, 0, 0, 0, 0, 0, 0
```

Obrázek 8: Ukázka struktury souboru typu ARFF

Každý soubor typu ARFF tedy obsahuje tři samostatné části. V první uvádíme název takového souboru.

V druhé části je uváděn výčet atributů, používaných nad těmito daty. A ve třetí části jsou uvedena samotná data. Všechny tyto tři části jsou uvozeny klíčovými slovy, které byly představeny v úvodu práce a jsou obsaženy i v ukázce nad tímto textem.
Důležitým poznatkem je to, že počet atributů se musí rovnat počtu prvků jednotlivých instancí. V opačném případě nástroj Weka upozorní, že tyto data není schopen zpracovat.

Pro začátek tedy bylo nutné spojit soubory typu FeatNames a EgoFeat. Spojení samostatných souborů pro jednotlivé sítě proběhlo v pořádku. Nicméně aby bylo možné tyto data dále lépe analyzovat, bylo nutné spojit všechny soubory typu FeatNames a EgoFeat pro jednotlivé sítě. Proto bylo nutné vytvořit seznam atributů nad celou datovou kolekcí. V jednotlivých souborech se tyto atributy lišily. Pro tuto činnost musel být vytvořen program v jazyce C#, který tuto činnost dokázal rychle provést. Celý tento program je přiložen na CD.

Při snaze takto upravená data, obsažená v jednom ARFF souboru, vložit do nástroje WEKA a začít zpracovávat ovšem vyvstal problém s nejedinečností atributů nad celou kolekcí. Vytvořený soubor obsahoval duplicity v rámci atributů, které znemožňovaly

další práci v nástroji Weka. Tento nástroj totiž pro svoji práci potřebuje, aby byly všechny atributy uvedené v testovacím souboru jednoznačné.

Bylo tedy nutné C# program poupravit, aby bylo možné se duplicitních atributů vyvarovat.

Po této úpravě už proběhlo vytvoření jediného ARFF soubor obsahujícího spojení souborů FeatNames a EgoFeat, který obsahoval veškeré data nad touto kolekcí a byl připraven pro další zpracovávání v nástroji Weka.

Při importu se navíc objevil problém s kódováním souboru. Bylo tedy nutné provést změnu kódování z UTF8 na ASCII. Takto připravený soubor však nebylo možné importovat do nástroje Weka. Bránila tomu hodnota nejvyšší používané paměti, která je nastavená ve spouštěcím skriptu tohoto nástroje. Po dosažení této hodnoty totiž dochází k ukončení výpočtu nástroje, a tudíž i k ukončení samotné operace importu či pozdějšího shlukování. Bylo tedy nutné hodnotu navýšit až na 2.900 MB. Poté již načtení souboru do nástroje WEKA nic nebránilo a proběhlo v pořádku.

Načtený soubor obsahoval více než 157.000 atributů a téměř 1.000 instancí. Pro prvotní testování byl vybrán algoritmus COBWEB. Při shlukovaní ovšem vznikl problém právě s pamětí. Pro takový počet atributů a instancí by totiž bylo nutné navýšit paměť ještě více, což ovšem nebylo technicky možné. Z toho důvodu bylo provedeno rozdělení datové kolekce na více souborů. Jako optimální se jevilo vytvořit více zdrojových souborů se zvyšujícím se počtem atributů a instancí. Aby bylo možné nalézt maximální možnou hodnotu pro testování na daném stroji. Po sérii pokusů bylo toto rozpětí stanoveno od 1.000 atributů až po 50.000 s krokem 10.000 (výjimkou byl krok s 5.000 atributy). Hodnota 50.000 atributů byla zároveň otestována jako nejvyšší možná pro použití při porovnávání shlukovacích algoritmů na testovacím NTB.

Po provedení těchto úprav již bylo možné začít testovat jednotlivé shlukovací algoritmy.

5.2 Porovnání zvolených algoritmů

V rámci porovnávání shlukovacích algoritmů v nástroji WEKA byl kladen důraz na tři základní hodnoty. V rámci měření probíhalo sledování využití procesoru a paměti při provádění shlukování a zároveň byl sledován čas potřebný pro výpočet těchto algoritmů.

Měření probíhalo pro jednotlivé algoritmy a pro jednotlivé soubory s počty atributů od nejmenšího po nejvyšší. Všechny tyto testovací soubory lze nalézt na přiloženém CD.

Veškeré výpočty byly prováděny na notebooku s procesorem Intel(R) Core(TM) i3-3227U CPU @ 1.90GHz 1.90GHz, pamětí 4,00 GB a na 64bitovém operačním systému Windows 8 s 64bitovou verzí nástroje WEKA. Proto jsou všechny procentuální hodnoty vztažené k těmto výchozím hodnotám.

5.2.1 COBWEB algoritmus

Při shlukování pomocí algoritmu COBWEB docházelo k navýšení vytížení procesoru v rozmezí 28,2% až 33,9%, které rostlo rovnoměrně při navyšování počtu zpracováva-

	1000	5000	10000	20000	30000	40000	50000
vytížení procesoru (%)	28,2	28,8	29,3	30,7	31,9	33,7	33,9
vytížení paměti (MB)	147	298	451	965	1107	1169	1253
výpočetní čas (s)	15,64	92,00	251,58	689,67	1167,45	1686,71	2395,90

Tabulka 1: Výčet zatížení procesoru, paměti a výpočetní čas algoritmu COBWEB

ných atributů a instancí. V rámci sledovaných hodnot došlo k nejvyššímu růstu zatížení procesoru při kroku pro 40.000 atributů.

V rámci využití paměti se pohybovala potřebná hodnota mezi 147 MB a 1253 MB. Docházelo k rovnoměrnému navyšování hodnoty používané paměti. Při zpracovávání 50.000 atributů však došlo k již výraznějšímu navýšení této hodnoty.

Výpočetní čas pro algoritmu COBWEB se zvyšoval taktéž rovnoměrně. Jediného většího výkyvu nastalo u počtu atributů 5.000, což bylo způsobeno právě polovičním nárůstem počtu atributů než v dalších krocích. Hodnota výpočetního času se pohybovala v rozmezí od 15,64 s po 2395,90 s.

Kromě sledování výše uvedených hodnot bylo nutné i porovnat výsledky shlukování v nástroji WEKA. Hodnoty stanovené pro jednotlivé kroky jsou uvedeny v tabulce níže.

Ve všech tabulkách týkajících se získaných dat se u počtu atributů vyskytují hodnoty rozšířené o 1. Je to z důvodu, že počet atributů byl rozšířen o atribut definující ID zpracovávané sítě.

Veškeré výstupní soubory obsahující podrobnosti k prováděnému shlukování jsou umístěny na přiloženém CD.

počet instancí	176	550	748	877	909	928	933
počet atributů	1001	5001	10001	20001	30001	40001	50001
počet spojení	0	0	0	0	0	0	0
počet rozdělení	0	0	0	0	0	0	0
počet shluků	1	1	1	1	1	1	1
počet instancí ve shluku	176	550	748	877	909	928	933
(%)	100	100	100	100	100	100	100

Tabulka 2: Získaná data o shlukování z nástroje WEKA k algoritmu COBWEB

Nástroj WEKA tedy v případě všech testovaných souborů nalezl pouze jeden shluk, který vždy obsahoval 100% vstupních atributů. Vizuální reprezentace pro jednotlivé kroky jsou obsaženy v příloze A na konci této práce. Tyto grafy jsou přiloženy pro všechny kroky všech sledovaných algoritmů.

5.2.2 DBSCAN algoritmus

Během shlukování pomocí algoritmu DBSCAN docházelo k navýšení vytížení procesoru v rozmezí 21,4% a 28,3%, které rostlo pomalu při navyšování počtu zpracovávaných

	1000	5000	10000	20000	30000	40000	50000
vytížení procesoru (%)	21,4	26,6	27,8	27,9	28,1	28,3	28,3
vytížení paměti (MB)	193	319	588	654	734	795	809
výpočetní čas (s)	0,83	33,32	127,37	586,91	1049,75	1493,43	2047,24

Tabulka 3: Výčet zatížení procesoru, paměti a výpočetní čas algoritmu DBSCAN

atributů a instancí. V rámci sledovaných hodnot došlo k nepatrně většímu nárůstu až od hodnoty 30.000 atributů.

Hodnota využívané paměti se pohybovala mezi 193 a 809 MB. Docházelo k rovnoměrnému navyšování hodnoty používané paměti. K největšímu navýšení hodnoty paměti došlo při zpracování 10.000 atributů. Což bylo způsobeno polovičním navýšením počtu zpracovávaných atributů.

Výpočetní čas u tohoto algoritmu se pohyboval v rozmezí 0,83 s a 2047,24 s. Stejně jako u hodnoty paměti došlo k výraznému navýšení výpočetního času u kroku 10.000 atributů.

počet instancí	176	550	748	877	909	928	933
počet atributů	1001	5001	10001	20001	30001	40001	50001
hodnota Epsilon	0,9	0,9	0,9	0,9	0,9	0,9	0,9
hodnota minPts	6	6	6	6	6	6	6
počet shluků	0	4	1	1	0	0	0
počet instancí	0	7, 6, 6, 6	18	9	0	0	0
ve shluku (%)	0	28, 24, 24, 24	100	100	0	0	0
počet instancí bez shluku	176	525	730	868	909	928	933

Tabulka 4: Získaná data o shlukování z nástroje WEKA k algoritmu DBSCAN

Při použití algoritmu DBSCAN se podařilo nástroji WEKA při kroku 5.000 atributů nalézt 4 shluky s počtem 7, 6, 6 a 6 instancí. Pro kroky 10.000 a 20.000 atributů byly taktéž nalezeny shluky. Vždy se jednalo o shluk jeden. V případě kroku pro 10.000 atributů obsahoval shluk 18 instancí a při kroku 20.000 obsahoval 9 instancí. Po všechny ostatní kroky tento algoritmus žádné další shluky nenašel a zbylé instance vyhodnotil jako instance bez přítomnosti ve shluku.

5.2.3 EM algoritmus

Při shlukování pomocí algoritmu EM se hodnota vytížení procesoru pohybovala mezi hodnotami 27,5 a 31,3 %. Největší navýšení této hodnoty proběhlo při výpočtu pro 30.000 atributů. Jinak probíhalo vytížení s mírnou vzestupnou tendencí.

Mezi hodnotou využívané paměti pro 1.000 atributů a tou samou hodnotou pro 50.000 atributů vznikl velký rozdíl. Pro první krok nástroj WEKA využíval 259 MB paměti. Pro krok 5.000 ovšem došlo již ke značnému navýšení hodnoty paměti až na úroveň 906 MB.

	1000	5000	10000	20000	30000	40000	50000
vytížení procesoru (%)	27,5	27,7	28,1	28,3	29,9	30,1	31,3
vytížení paměti (MB)	259	906	1093	1390	1408	1706	2184
výpočetní čas (s)	79,73	420,04	1199,44	3687,59	5611,30	7485,26	9899,66

Tabulka 5: Výčet zatížení procesoru, paměti a výpočetní čas algoritmu EM

Dále nastal zlom u kroku pro 30.000 atributů, ve kterém došlo k zastavení takto prudkého vzestupu hodnoty používané paměti. Pro další kroky ovšem paměť opět narůstala rychleji.

Výpočetní čas pro algoritmus EM se pohyboval ve velmi velkém rozpětí. Tato skutečnost je způsobena složitým průběhem výpočtu tohoto algoritmu. Pro první krok byl výpočetní čas 79,73 vteřin. Nicméně již od druhého kroku docházelo k velmi výraznému navýšení času. Pro krok 50.000 již tedy nástroj WEKA potřeboval 9899,66 s. Což je nejvyšší hodnota ze všech sledovaných algoritmů.

počet instancí	176	550	748	877	909	928	933
počet atributů	1001	5001	10001	20001	30001	40001	50001
počet shluků	4	1	1	1	1	1	1
počet instancí ve shluku (%)	8, 16, 152 5, 9, 86	550 100	748 100	877 100	909 100	928 100	933 100

Tabulka 6: Získaná data o shlukování z nástroje WEKA k algoritmu EM

V případě použití algoritmu EM byl schopen nástroj WEKA zjistit již v prvním kroku, že existují 4 shluky. Počet instancí v nich byl 8, 16 a 152, což odpovídá 5, 9 a 86%. U dalších kroků algoritmus EM již pouze definoval vždy jeden shluk, který obsahoval všechny použité instance.

5.2.4 FARTHEST FIRST algoritmus

	1000	5000	10000	20000	30000	40000	50000
vytížení procesoru (%)	20,3	27,3	31,5	42,4	45,4	47,2	48,3
vytížení paměti (MB)	166	331	538	795	1036	1156	1834
výpočetní čas (s)	0,11	0,33	0,67	2,06	3,49	4,69	6,48

Tabulka 7: Výčet zatížení procesoru, paměti a výpočetní čas algoritmu FARTHEST FIRST

Během procesu shlukování pomocí algoritmu FARTHEST FIRST došlo k velmi výraznému zvýšení vytížení procesoru. Pro první krok tento algoritmus potřeboval velmi nízkou hodnotu procesoru, jednalo se o 20,3 MB. Nicméně již od druhého kroku došlo ke zvýšení potřeby vytížení procesoru. Největší navýšení však proběhlo u kroku pro 20.000 atributů. Poté i dále docházelo k velkému navýšení při každém kroku až pro 50.000 atributů, kdy bylo potřeba 48,3% vytížení procesoru počítače. Což se jedná o nejvyšší hodnotu ze všech porovnávaných algoritmů. Druhá nejvyšší hodnota zatížení procesoru je 33,9%

pro algoritmus COBWEB. Jde tedy jasně vidět, jak výrazný nárůst u tohoto algoritmu proběhl.

U využití paměti došlo taktéž ke značnému navýšení mezi prvním a posledním krokem. Nicméně toto navýšení není tak výrazné jako u zatížení procesoru. U algoritmu FARTHEST FIRST se hodnota paměti pohybovala v rozmezí 166 a 1834 MB. Hodnota rosta rovnoměrně v závislosti na počtu zpracovávaných atributů. Největšího navýšení bylo potřeba až u kroku pro 50.000 atributů.

Výpočetní čas pro algoritmus FARTHEST FIRST byl ovšem pozorován nejnižší ze všech sledovaných algoritmů. Pohyboval se od 0,11 do 6,48 s. Hodnota pro 50.000 atributů je tak malá, že je například nižší než hodnota výpočetního času pro algoritmy COBWEB a EM pro krok 1.000.

počet instancí	176	550	748	877	909	928	933
počet atributů	1001	5001	10001	20001	30001	40001	50001
počet shluků	2	2	2	2	2	2	2
počet instancí ve shluku (%)	174, 2 99, 1	549, 1 100, 0	747, 1 100, 0	876, 1 100, 0	908, 1 100, 0	927, 1 100, 0	932, 1 100, 0

Tabulka 8: Získaná data o shlukování z nástroje WEKA k algoritmu FARTHEST FIRST

V rámci hledání shluků byly ve všech případech nalezeny 2 shluky. V rámci kroku pro 1.000 atributů nalezl shluky v počtu 174 a 2 instance. Od dalších kroků bylo vždy možné identifikovat vždy 2 shluky, kdy jeden obsahoval jednu instanci a druhý všechny zbylé.

5.2.5 K-MEANS algoritmus

	1000	5000	10000	20000	30000	40000	50000
vytížení procesoru (%)	29,5	29,5	29,7	29,9	30,5	31,4	32,2
vytížení paměti (MB)	195	498	502	947	1183	1424	1895
výpočetní čas (s)	0,28	1,89	5,99	16,22	27,54	34,85	45,14

Tabulka 9: Výčet zatížení procesoru, paměti a výpočetní čas algoritmu K-MEANS

Při shlukování algoritmem K-MEANS se zatížení procesoru pohybovalo mezi hodnotami 29,5 a 32,2 %. Zatížení rostlo rovnoměrně se zvyšujícím se počtem atributů a instancí. K většímu nárůstu této hodnoty došlo během kroku pro 30.000 atributů.

Využití paměti během výpočtu tohoto algoritmu se pohybovalo v rozmezí 195 až 1895 MB. V rámci nárůstu této hodnoty došlo k velmi zajímavému jevu u kroku pro 5.000 a 10.000 atributů. Pro tyto dvě hodnoty dosahovala velikost využívané paměti hodnot 498 a 502 MB. Rozdíl mezi těmi hodnotami je tedy pouze 4 MB. Další nárůst probíhal již rovnoměrně. U kroku 50.000 navíc ale došlo ke zrychlení nárůstu této hodnoty.

Výpočetní čas pro algoritmus K-MEANS se pohyboval v rozmezí 0,28 a 45,14 s. Čas nutný pro výpočet vzrost dle počtu zpracovávaných atributů. S nejvyšší dosaženou hodnotou 45,14 s je druhý nejrychlejší ze všech sledovaných algoritmů.

počet instancí	176	550	748	877	909	928	933
počet atributů	1001	5001	10001	20001	30001	40001	50001
počet shluků	2	2	2	2	2	2	2
počet instancí ve shluku	28, 148	1, 549	276, 633	872, 5	908, 1	1, 927	1, 932
(%)	16, 84	0, 100	0, 100	99, 1	30, 70	0, 100	0, 100

Tabulka 10: Získaná data o shlukování z nástroje WEKA k algoritmu K-MEANS

Při použití shlukovacích algoritmu K-MEANS byl nástroj WEKA schopen pro každý krok identifikovat dva shluky. Za povšimnutí stojí pořadí atributů v jednotlivých shlucích. Jednotlivé shluky tedy obsahují 18, 148; 1, 549; 276, 633; 872, 5; 908, 1; 1, 927 a 1, 932 atributů.

5.2.6 OPTICS algoritmus

	1000	5000	10000	20000	30000	40000	50000
vytížení procesoru (%)	25,2	27,8	27,9	27,9	27,9	28,5	29,2
vytížení paměti (MB)	210	321	479	575	634	658	969
výpočetní čas (s)	1,03	32,9	129,84	596,20	1047,29	1489,84	2132,28

Tabulka 11: Výčet zatížení procesoru, paměti a výpočetní čas algoritmu OPTICS

Během průběhu výpočtu shlukování algoritmu OPTICS se pohybovala hodnota zatížení procesoru mezi hodnotami 25,2 a 29,2%. Pro kroky 10.000, 20.000 a 30.000 dokonce došlo ke stejnému průměrnému zatížení procesoru a to na hodnotě 27,9 MB. Poté došlo ke zvýšení zatížení. U žádného z pozorovaných algoritmů nedošlo k takovéto situaci. Dokonce i hodnota pro krok 5.000 atributů je velmi blízko dané hodnotě.

Vývoj využití paměti probíhalo rovnoměrně s rostoucím počtem atributů a instancí. Mezi kroky 20.000 a 40.00 došlo dokonce ke zpomalení růstu hodnoty využívané paměti. V rámci kroku 50.000 nicméně došlo ke značnému navýšení hodnoty využívané paměti až na hodnotu 969 MB. I tak se ovšem jedná o druhou nejnižší hodnotu pro tento krok v rámci všech porovnávaných algoritmů.

Výpočetní čas pro algoritmus OPTICS více méně kopíruje průběh výpočetního času pro algoritmus DBSCAN, což je způsobeno, že je z něj odvozen. Od kroku 10.000 došlo ke zrychlení nárůstu hodnoty výpočetního času. Hodnota od kroku 10.000 byla ale rostoucí rovnoměrně se zvyšujícím se počtem atributů a instancí.

Z tabulky tedy vyplývá, že algoritmus OPTICS pro tyto data nebyl schopen v žádném kroku nalézt shluky.

5.2.7 Vzájemné porovnání sledovaných algoritmů

Po vyhodnocení získaných informací ze shlukové analýzy jednotlivých algoritmů nám vyplývají níže popsané skutečnosti, které vždy popisují algoritmy s nejvyšší a nejnižší

počet instancí	176	550	748	877	909	928	933
počet atributů	1001	5001	10001	20001	30001	40001	50001
hodnota Epsilon	0,9	0,9	0,9	0,9	0,9	0,9	0,9
hodnota minPts	6	6	6	6	6	6	6
počet shluků	0	0	0	0	0	0	0
počet instancí ve shluku (%)	0 0	0 0	0 0	0 0	0 0	0 0	0 0
počet instancí bez shluku	176	550	748	877	909	928	933

Tabulka 12: Získaná data o shlukování z nástroje WEKA k algoritmu OPTICS

danou hodnotou. Tato hodnota je vztažená k počáteční hodnotě pro 1.000 atributů a ke konečné pro 50.000 atributů. U jednotlivých algoritmu vždy záleží, pro jaký počet mají být použity. Od toho počtu se pak dále liší výsledné pořadí algoritmů.

Porovnání algoritmů podle vytížení procesoru

Vytížení procesoru při výpočtu algoritmu FARTHEST FIRST začínalo na nejnižší hodnotě ze všech sledovaných algoritmů. Nicméně pro poslední krok dosahovalo zároveň i nejvyšší hodnoty ze všech porovnávaných algoritmů. Nárůst hodnoty proběhl z 20,3% na 48,3%. Hodnota vytížení procesoru pro tento algoritmus zároveň během všech kroků rostla nejprudčeji.
Naopak u algoritmů DBSCAN a OPTICS bylo vytížení procesoru nižší. V případě algoritmu DBSCAN začínalo na hodnotě 21,4% a končilo na 28,3%. U algoritmu OPTICS začínalo na hodnotě 25,2% a končilo na 29,2%. U algoritmu DBSCAN ovšem došlo během kroku pro 5.000 atributů k výraznému nárustu této hodnoty. Ve všech ostatních případech rostla tato hodnota u obou algoritmů rovnoměrně v závislosti na počtu atributů a instancí.

Grafické vyjádření vývoje hodnoty zatížení procesoru během výpočtu všech sledovaných algoritmů se nachází v grafu níže v textu této práce.

Porovnání algoritmů podle vytížení paměti

Nejvyšší hodnoty potřebné paměti pro výpočet u jednotlivých algoritmů bylo dosaženo při použití algoritmu EM. U tohoto algoritmu byla již počáteční hodnota nejvyšší ze všech sledovaných algoritmů. Vystoupala z 259 na 2184 MB.
Nejnižší koncové hodnoty využití paměti dosáhly algoritmy DBSCAN a OPTICS. Počáteční potřeba paměti sice nebyla u těchto dvou algoritmů nejnižší, nicméně konečná hodnota vyrostla u algoritmu DBSCAN na hodnotu 809 MB a u algoritmu OPTICS na hodnotu 969 MB. Což je v porovnání s algoritmem EM podstatný rozdíl.

Grafické vyjádření vývoje hodnoty využití paměti během výpočtu všech sledovaných algoritmů se nachází v grafu níže v textu této práce.

Porovnání algoritmů podle výpočetního času

Nejvyšší čas potřebný pro výpočet algoritmu byl pro algoritmus EM. Jednalo se o podstatný rozdíl mezi všemi ostatními algoritmy. Pro krok pro 1.000 atributů již dosahoval nejvyšší hodnoty ze všech sledovaných algoritmů. Konečná hodnota pro výpočetní čas u tohoto algoritmu dosáhla 9899,66 s. Pro ilustraci je nutné uvést druhý nejvyšší výpočetní čas. Toho bylo dosaženo pomocí algoritmu COBWEB, který dosahoval hodnoty 2395,90 s. Jde tedy vidět, že rozdíl je opravdu velmi výrazný.

Naopak nejnižšího výpočetního času bylo dosaženo algoritmem FARTHEST FIRST. Už pro první krok dosahoval výpočetní čas u tohoto algoritmu nejnižší hodnoty 0,11 s. Pro krok 50.000 dosahovala hodnota 6,48 s. Podobným způsobem proběhl nárust výpočetního času i u algoritmu K-MEANS, jehož výpočetní čas pro krok 50.000 atributů dosahoval 45,14 s.

Jelikož byla hodnota u algoritmu EM natolik vysoká, že zkreslovala zobrazení výsledku zbývajících algoritmů, bylo nutné algoritmus EM z grafu vyjmout, aby byl zřejmý nárust této hodnoty i u dalších algoritmů. Oba grafy zobrazující vyjádření vývoje hodnoty výpočetního času během výpočtu všech sledovaných algoritmů se nachází níže v textu této práce.

Porovnání algoritmů podle počtu nalezených shluků

V rámci porovnávání vybraných shlukovacích algoritmů však kromě výše uvedených třech sledovaných hodnot musí být také brán zřetel na počet nalezených shluků.

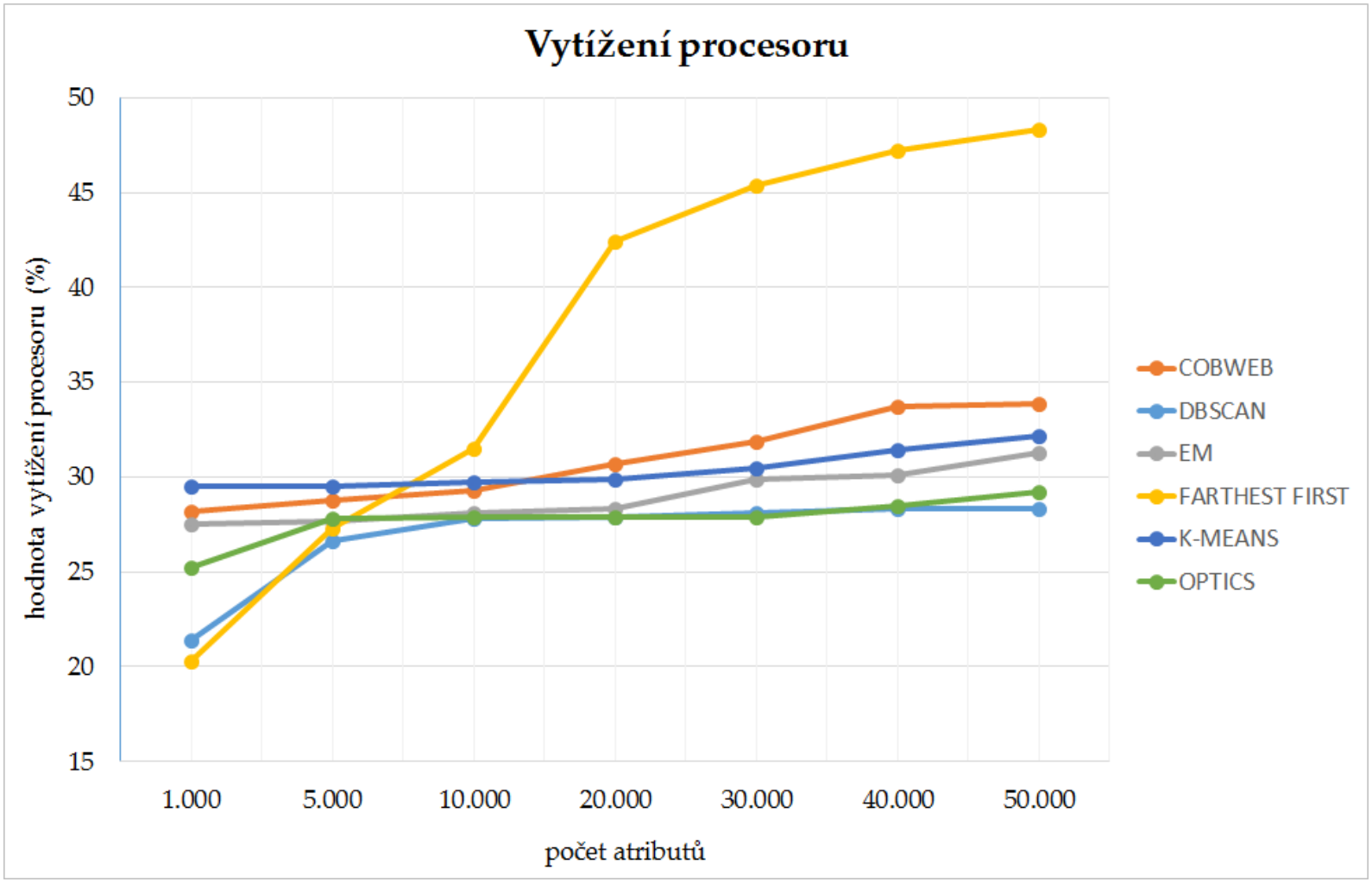
Algoritmus COBWEB byl pro všechny kroky schopen nalézt vždy jeden shluk, obsahující všechny instance.

U algoritmu DBSCAN byly objeveny shluky pro kroky 5.000, 10.000 a 20.000, pro všechny ostatní tento algoritmus nebyl schopnen žádné shluky nalézt, proto byly zbylé hodnoty označeny za hodnoty bez členství ve shluku.

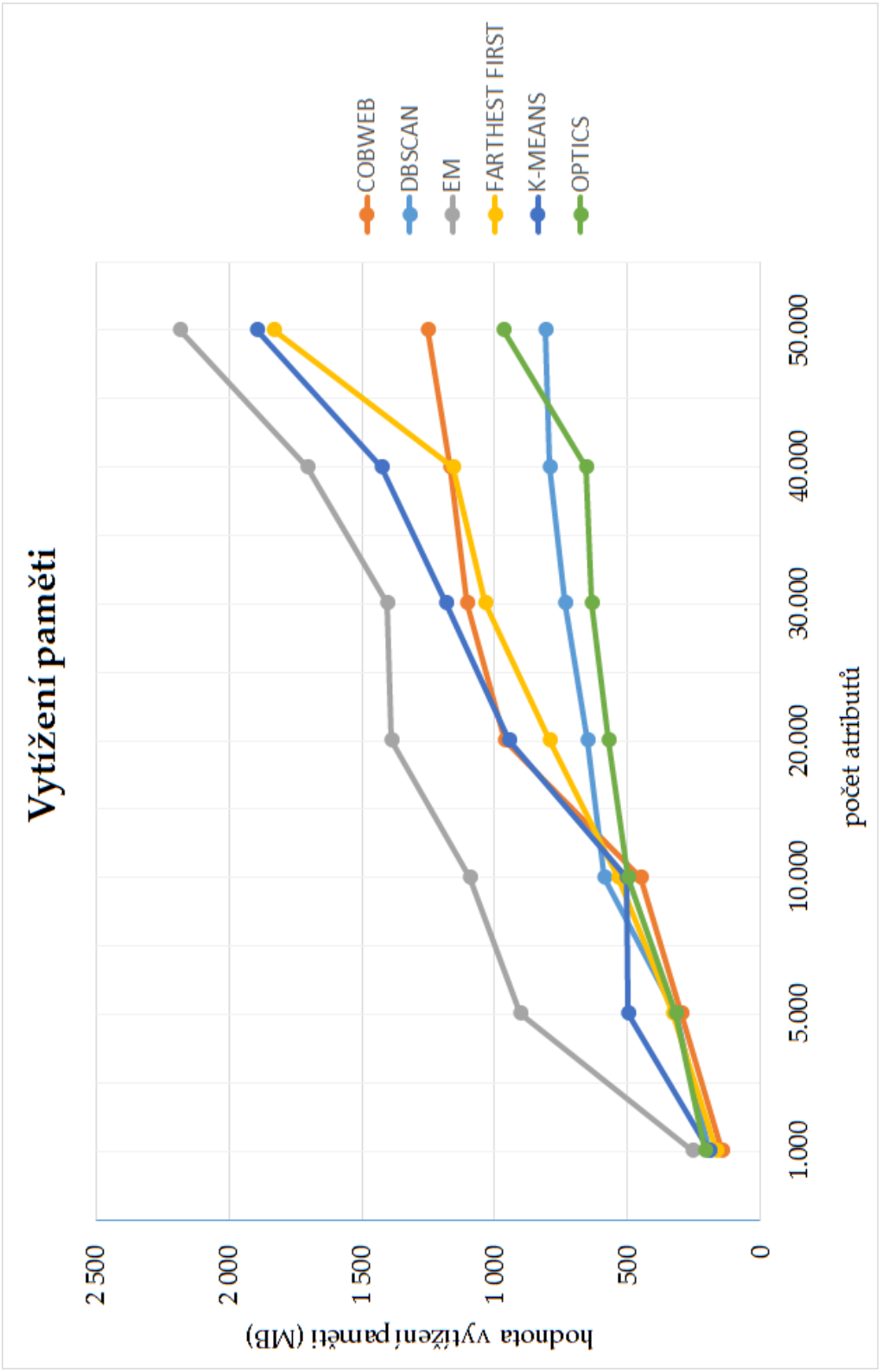
Algoritmus EM objevil shluky pouze u kroku 1.000. U všech ostatních kroků byl vždy nalezen pouze jeden shluk, obsahující všechny zbylé hodnoty.

Algoritmy FARTHEST FIRST a K-MEANS nalezly pro všechny kroky 2 shluky, které se lišily v počtu instancí do nich přiřazených.

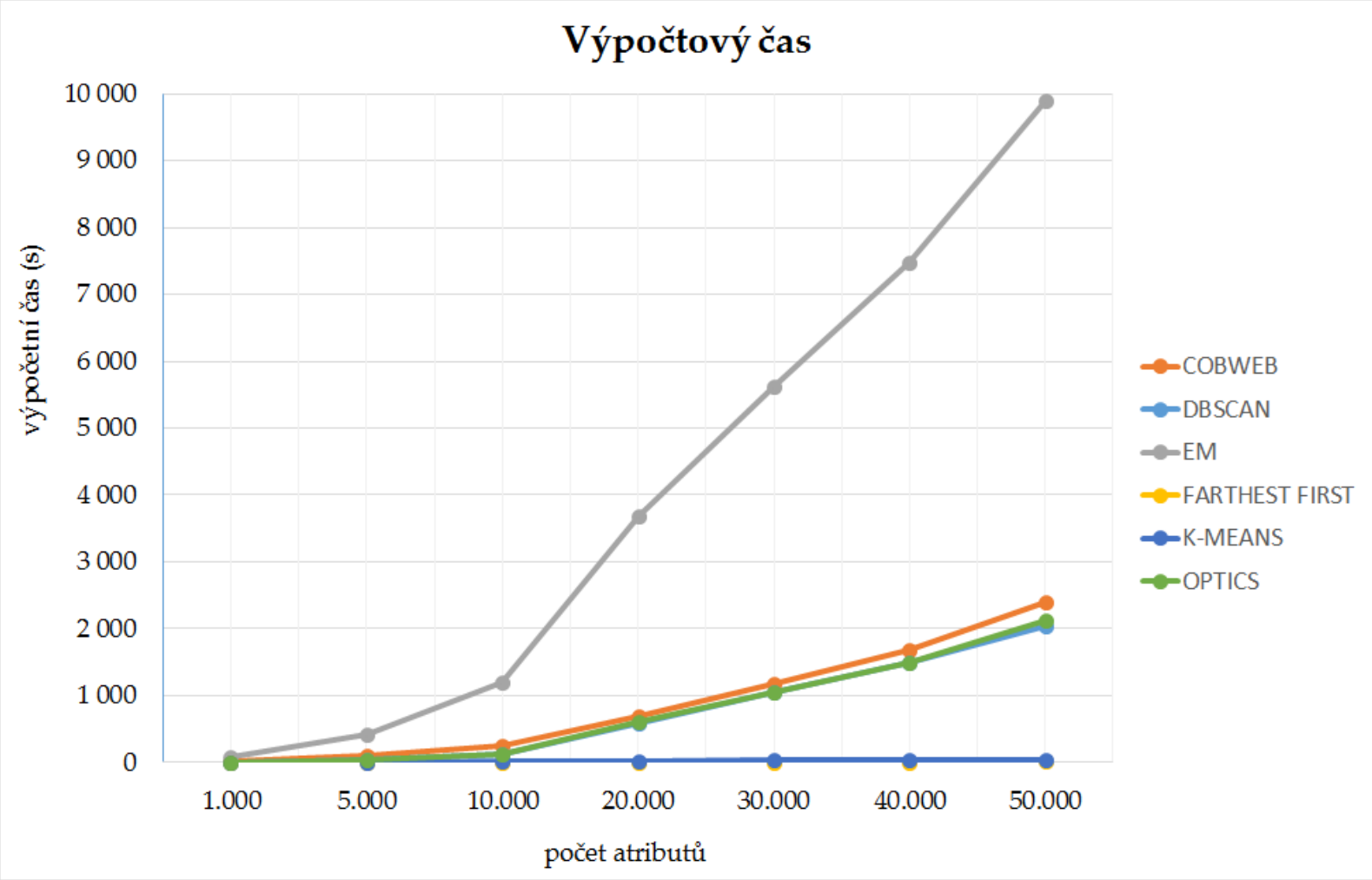
Algoitmus OPTICS nebyl schopen objevit žádné shluky ani pro jeden krok. Proto všechny hodnoty označil za hodnoty bez přitomnosti ve shluku.



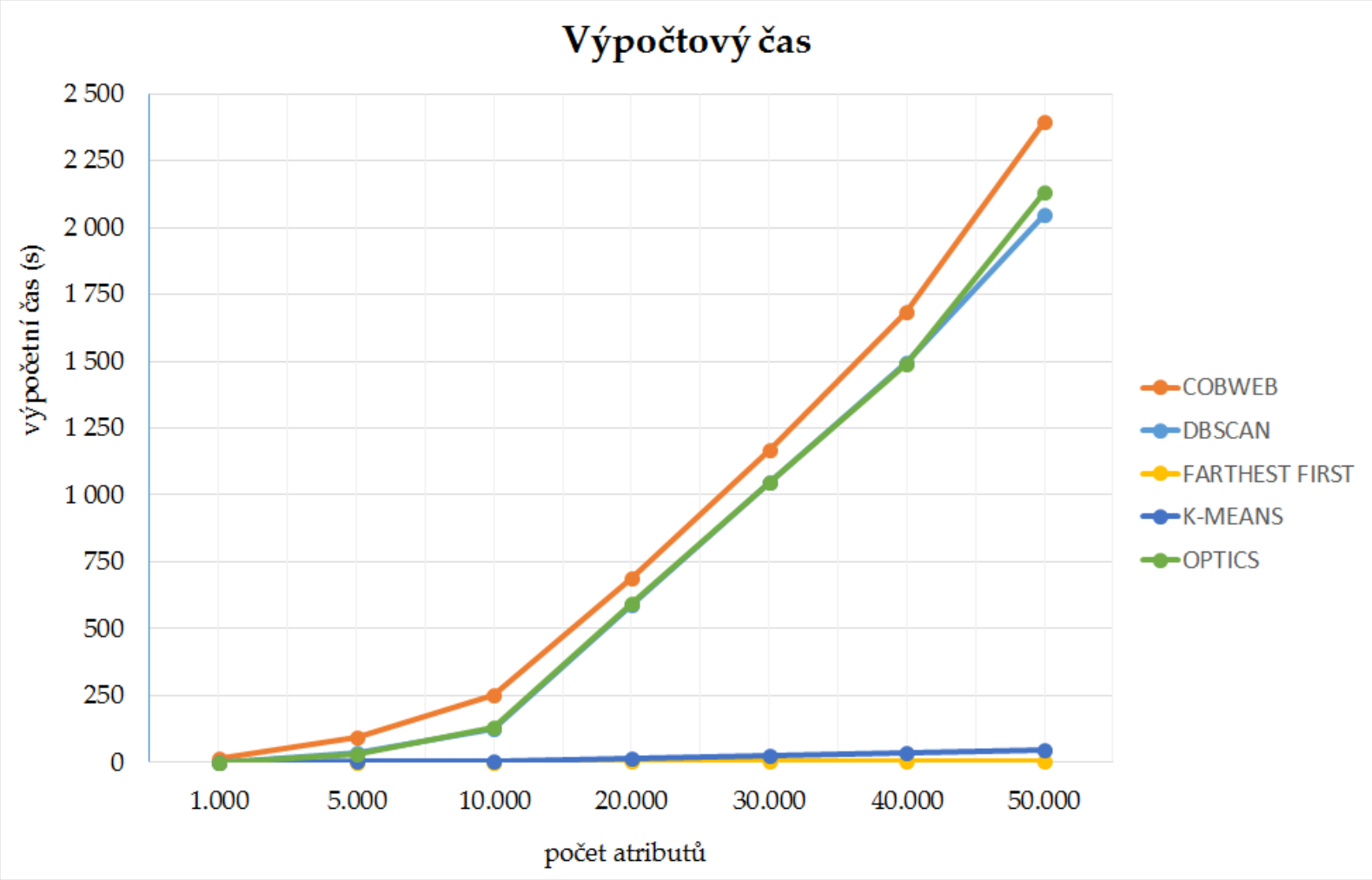
Obrázek 9: Porovnání sledovaných algoritmů v rámci hodnoty zatížení procesoru



Obrázek 10: Porovnání sledovaných algoritmů v rámci hodnoty potřebné paměti



Obrázek 11: Porovnání sledovaných algoritmů v rámci hodnoty výpočetního času



Obrázek 12: Porovnání sledovaných algoritmů v rámci hodnoty výpočetního času bez hodnoty pro algoritmus EM

6 Závěr

V průběhu této bakalářské práce proběhlo představení metod shlukové analýzy a jejich rozdělení. Zároveň byly podrobněji popsány vybrané shlukovací algoritmy. Pro jejich zpracování byl určen nástroj WEKA, který vznikl právě za účelem potřeby vytvoření nástroje pro tzv. dolování dat z rozlehlých sítí. Pro tuto práci byly jako výchozí data určena ta z programu Stanfordské univerzity, která byla získána právě pro tyto účely. Data se týkala sociálních sítí, proto bylo nutné se i v této práci věnovat problematice sociálních sítí. Byly představeny základní druhy a principy sociálních sítí. Podrobněji proběhlo představení sociální sítě Twitter, ze které byla vybrána testovavací data. Aby bylo ovšem možné tyto data zpracovávat, musela být upravena. Pro lepší pozorování nárustu výpočetního času, paměti a vytížení procesoru, bylo nutné vytvořit soubory pro různé počty atributů a instancí. Poté již bylo možné data otestovat a pozorovat rozdíly mezi jednotlivými algoritmy v návaznosti právě na výpočetní čas, pamět, využití procesoru i počet nalezených shluků. V závěru práce byla sesumarizována získaná data pro jednotlivé sledované parametry a byly vybrány vždy nejlepší a nejhorší algoritmy pro tyto znaky. Na závěr je však nutné podotknout, že není možné tyto algoritmy podrobněji porovnat. Vždy totiž záleží pro jak velká data mají být použita. Dále i záleží na tom, jakému parametru se dává přednost. Zda je potřeba algoritmus, který co nejrychleji vypočítá shluky v datech, či zda záleží na hodnotě využití procesoru nebo paměti. Tyto hodnoty se navíc liší i pro jednotlivé kroky. Pokud by bylo nutné docílit jasného porovnání, musel by být určen počet dat, vytížení procesoru, paměti a výpočetní čas pro jednotlivé hodnoty.

Jako hlavní přínos této práce považuji to, že jsem díky ní mohl více nahlédnout do problematiky shlukování a více se seznámit s jednotlivými shlukovacími algoritmy. Dále, že jsem také získal představu o hlavních výhodách či nevýhodách daných algoritmů. A mohl se naučit pracovat s nástrojem, který je určen pro dolování dat právě z rozlehlých sítí. Kromě této oblasti jsem se i více seznámil se sociálními sítěmi či jejich analýzou. Zároveň jsem se více zaměřil na sociální síť Twitter a zjistil principy, na kterých funguje. Na závěr je tedy nutné dodat, že během zpracování této práce jsem v oblasti shlukovacíh metod a algoritmů rozšířil znalosti, které velmi rád případně použiji i v praxi či dalším studiu.

7 Reference

[1] Sharma NARENDRA, Aman BAJPAI a Mr. Ratnesh LITORIZA. Comparison the various clustering algorithms of weka tools. [online]. 2012, Volume 2, Issue 5, s. 73-80 [cit. 28.9.2013].
Dostupné z: $http://www.ijetae.com/files/Volume2Issue5/IJETAE_0512_13.pdf$

[2] The University of Waikato in New Zealand. WEKA User Manual. [online]. 2012, [cit. 7.10.2013].
Dostupné z: $http://www.cs.waikato.ac.nz/ml/weka/documentation.html$

[3] Lukasová Alena, Šarmanová Jana, Metody shlukovací analýzy, SNTL Praha, 1985

[4] Kelbel Jan, Šilhán David, Shluková analýza [online]. [cit. 10.11.2013].
Dostupné z: $http://www.fd.cvut.cz/personal/nagyivan/Projekty/Classification/ShlukovaAnalyza.pdf$

[5] Douglas H. FISHER. Knowledge Acquisition Via Incremental Conceptual Clustering. Kluwer Academic Publisher, Boston [online]. 1987 [cit. 14.11.2013].
Dostupné z: $http://axon.cs.byu.edu/martinez/classes/678/Papers/Fisher_Cobweb.pdf/$

[6] G. Gan, C. Ma, and J. Wu. Data Clustering Theory, Algorithms and Applications. ASASIAM, 2007.

[7] Ranka Sanjay, Computer and Information Science and Engineering, University of Florida. Clustering, Part 5, 2008. [cit. 4.12.2013].
Dostupné z: $http://www.cise.ufl.edu/class/cis4930sp09dm/notes/dm5part5.pdf$

[8] Chuong B Do, Serafim Batzoglou, What is the expectation maximization algorithm? Nature Publishing Group, 2008. [cit. 27.1.2014].
Dostupné z: $http://ai.stanford.edu/chuongdo/papers/em_tutorial.pdf$

[9] Joe LATKO. Cluster Analysis in Supply Chain Optimization. [online]. 2012 [cit. 18.2.2014].
Dostupné z: $http://www.profitpt.com/joe-litko/cluster-analysis-in-supply-chain-optimizat/$

[10] Christopher Potts. Analysis: Clustering words by tags in the SwDA. University of Colorado at Boulder. [online]. 2013 [cit. 25.2.2014].
Dostupné z: $http://compprag.christopherpotts.net/swda-clustering.html$

[11] Kanungo Tapas, M. Mount David, S. Netanyahu Nathan, D. Piatko Christine, Silverman Ruth, Y. Wu Angela, An Efficient k-Means Clustering Algorithm: Analysis and Implementation, 2002 [online]. [cit. 2.3.2014].
Dostupné z: $https://www.cs.umd.edu/mount/Projects/KMeans/pami02.pdf$

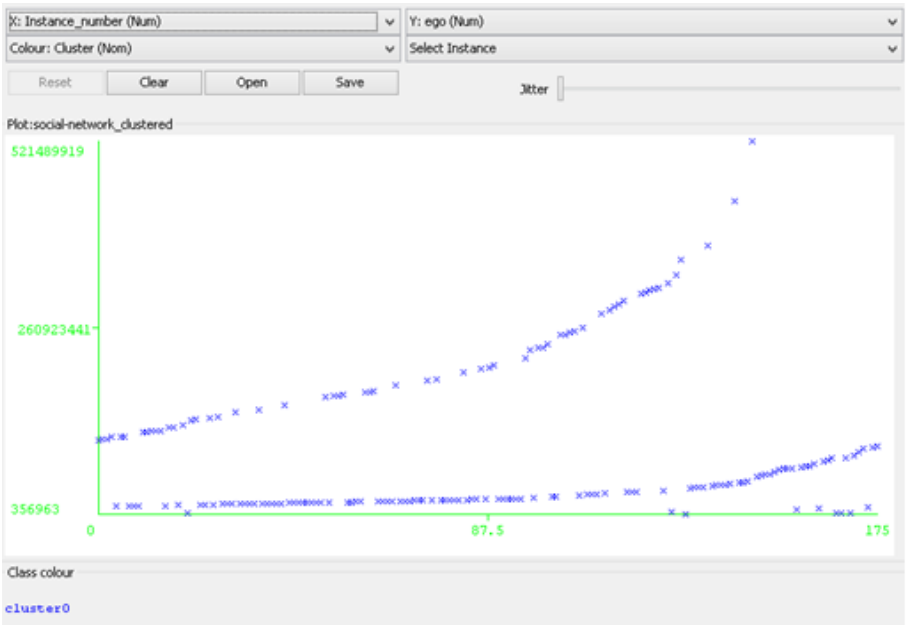
[12] Ian H. Witten, Elbe FRANK, Mark A. HALL, Data Mining: Practical Machine Learning Tool and Techniques. 3rd es. Morgan Kaufmann, 2011. ISBN 978-0-12-374856-0.

[13] Wasserman Stanley, Faust Katherine. Social Network Analysis in the Social and Behavioral Sciences. Cambridge University Press. 1994. ISBN 9780521387071.

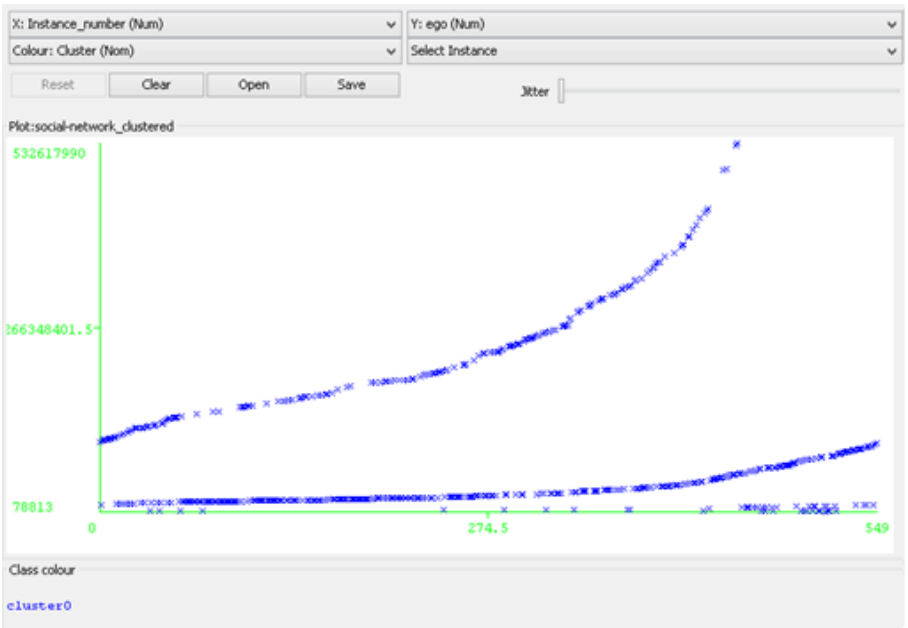
[14] Pinheiro Carlos. Social Network Analysis in Telecommunications. 2001 ISBN 978-1-118-01094-5.

[15] The University of Waikato in New Zealand. Attribute-Relation File Format (ARFF). [online]. 2008, [cit. 25.3.2014].
Dostupné z: $http://www.cs.waikato.ac.nz/ml/weka/arff.html$

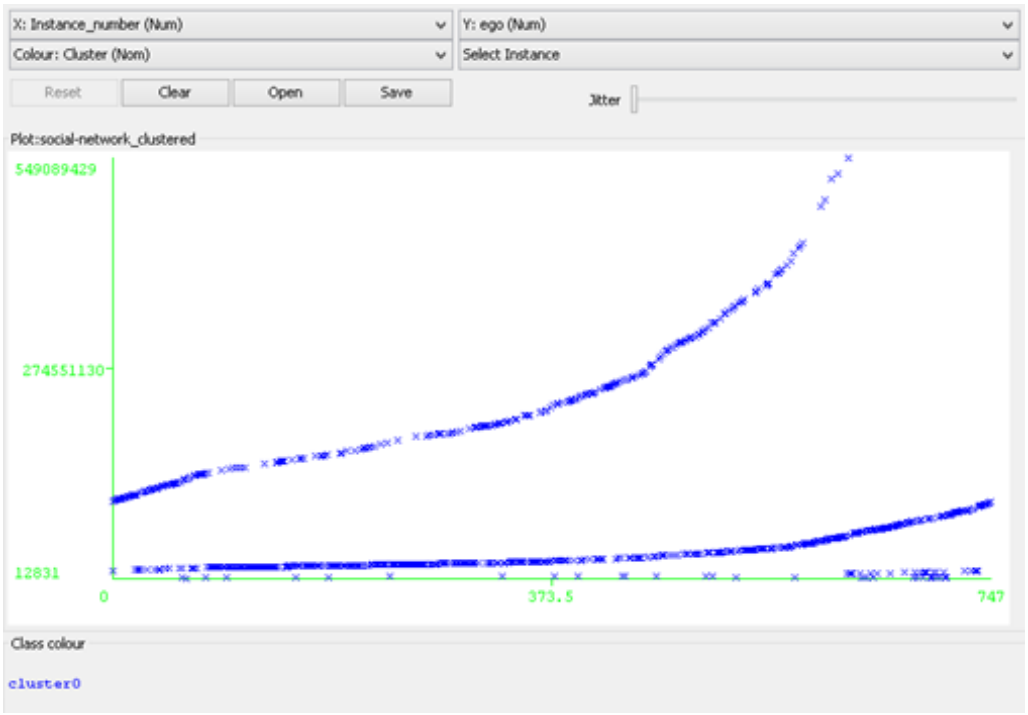
A Výstupní grafy pro jednotlivé algoritmy a počty atributů



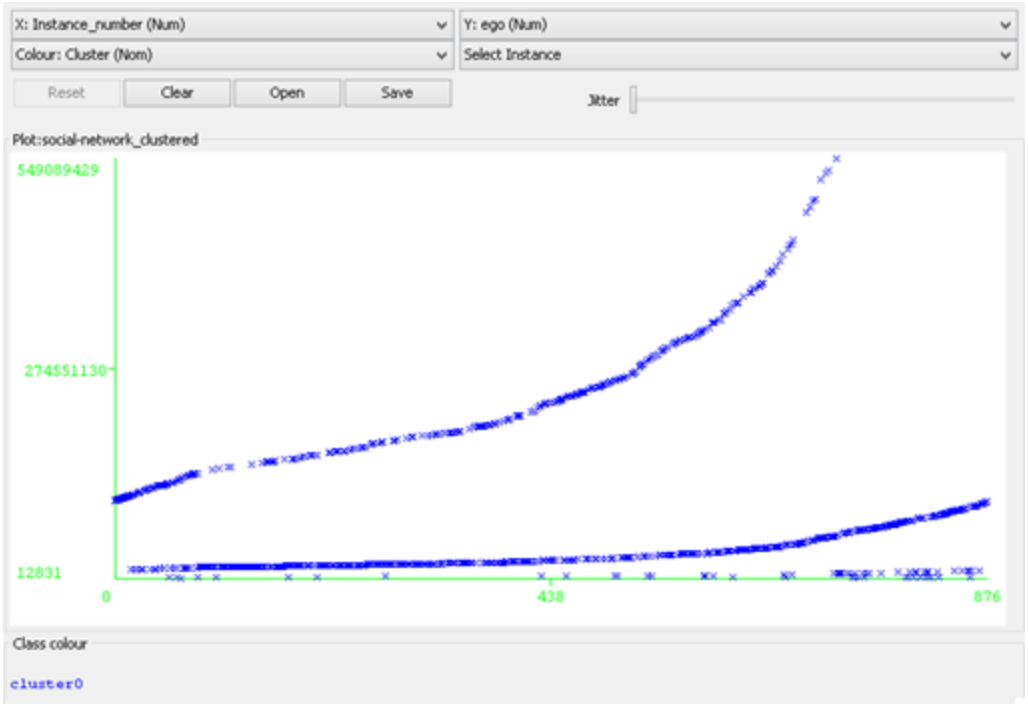
Obrázek 13: Výstupní graf pro 1.000 atributů - COBWEB



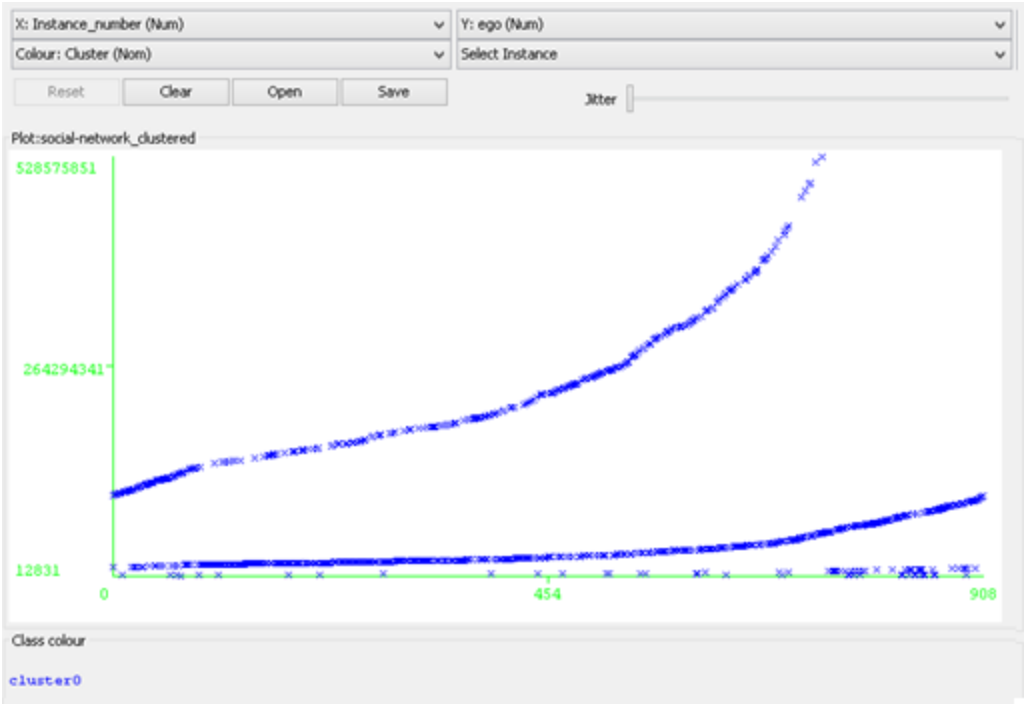
Obrázek 14: Výstupní graf pro 5.000 atributů - COBWEB



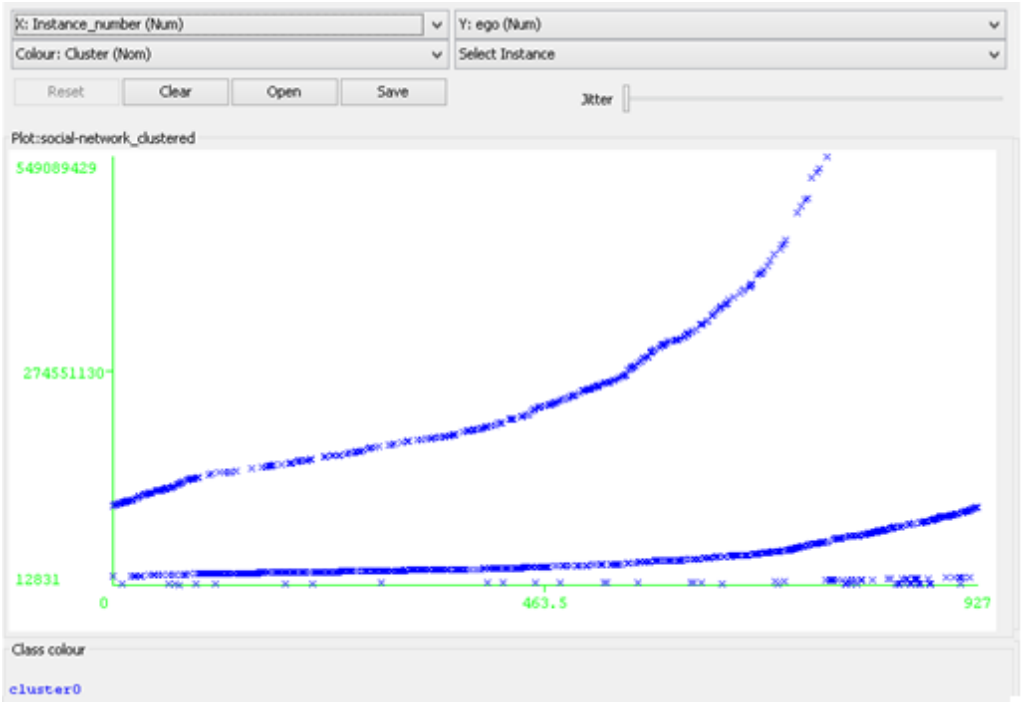
Obrázek 15: Výstupní graf pro 10.000 atributů - COBWEB



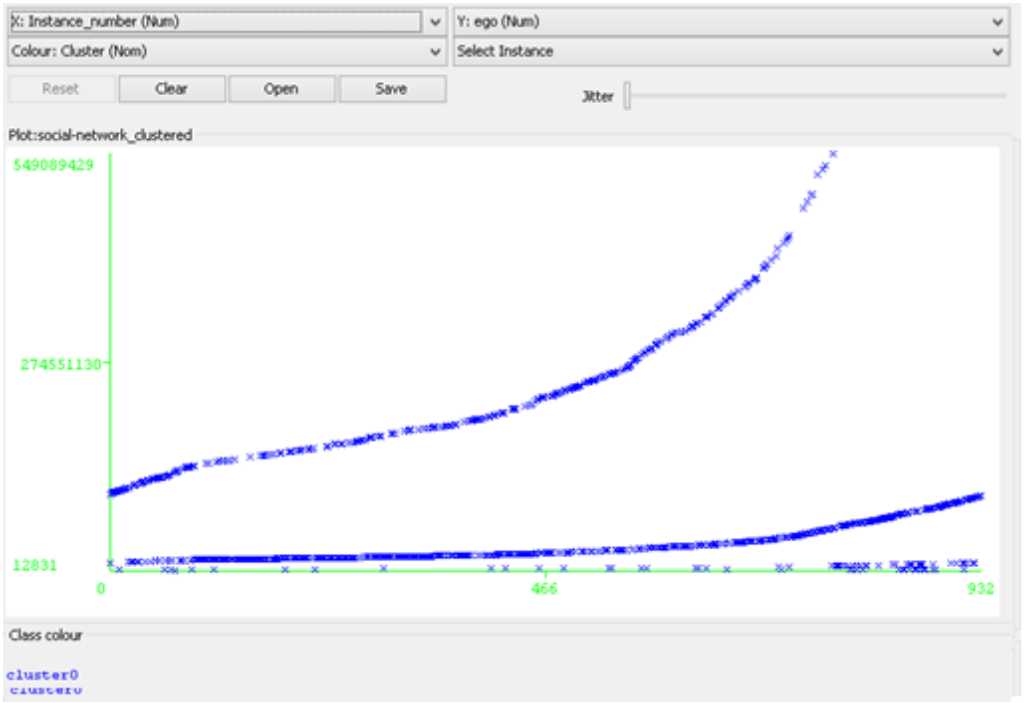
Obrázek 16: Výstupní graf pro 20.000 atributů - COBWEB



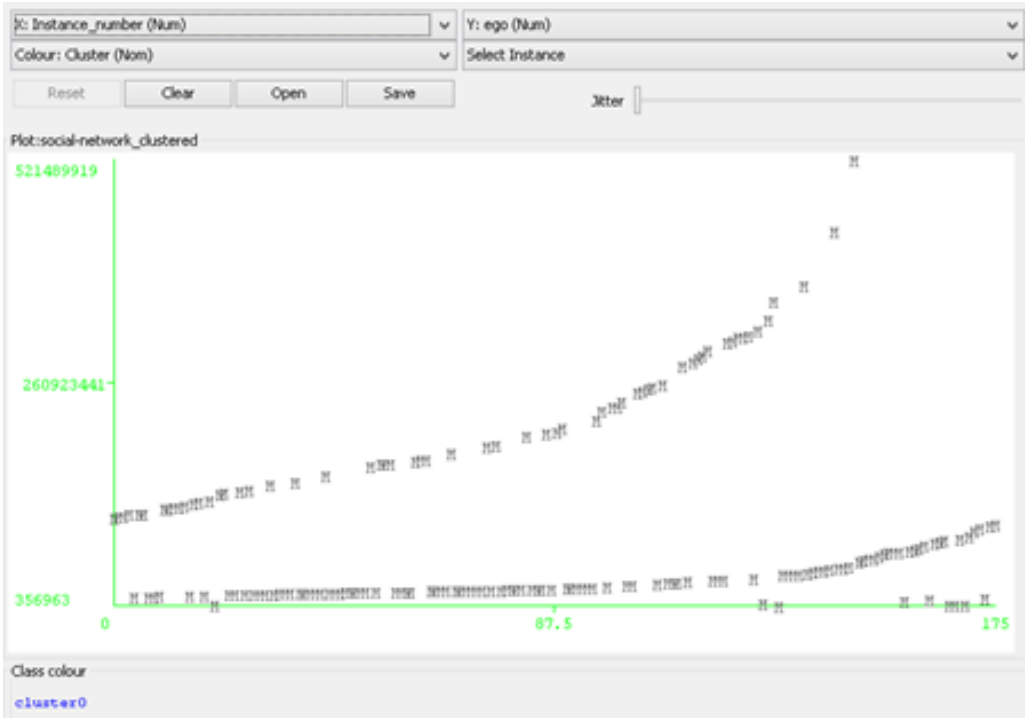
Obrázek 17: Výstupní graf pro 30.000 atributů - COBWEB



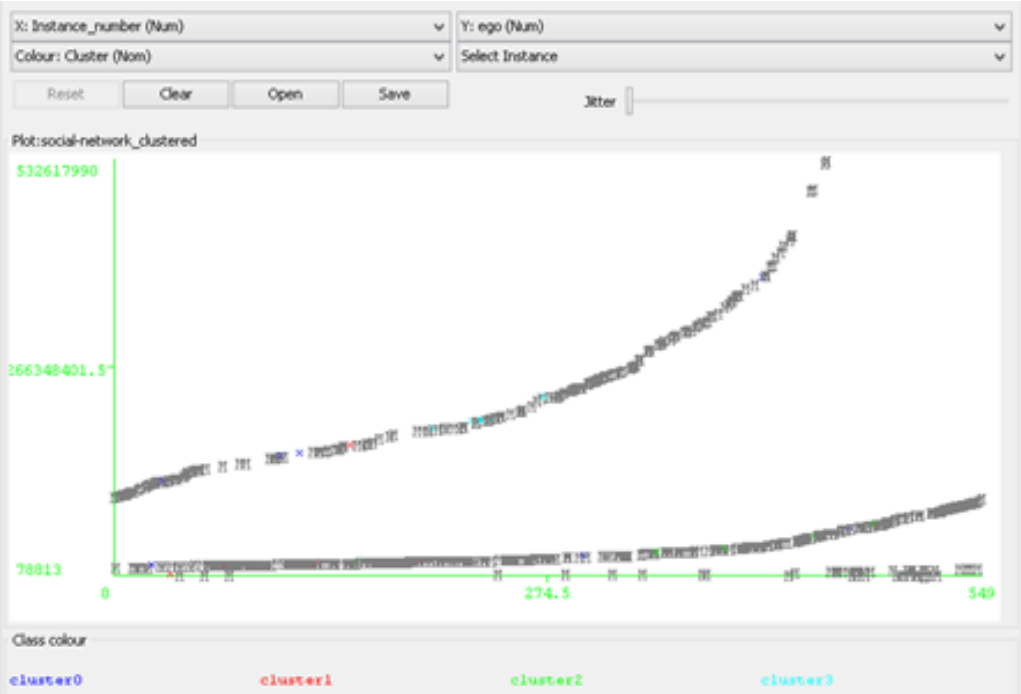
Obrázek 18: Výstupní graf pro 40.000 atributů - COBWEB



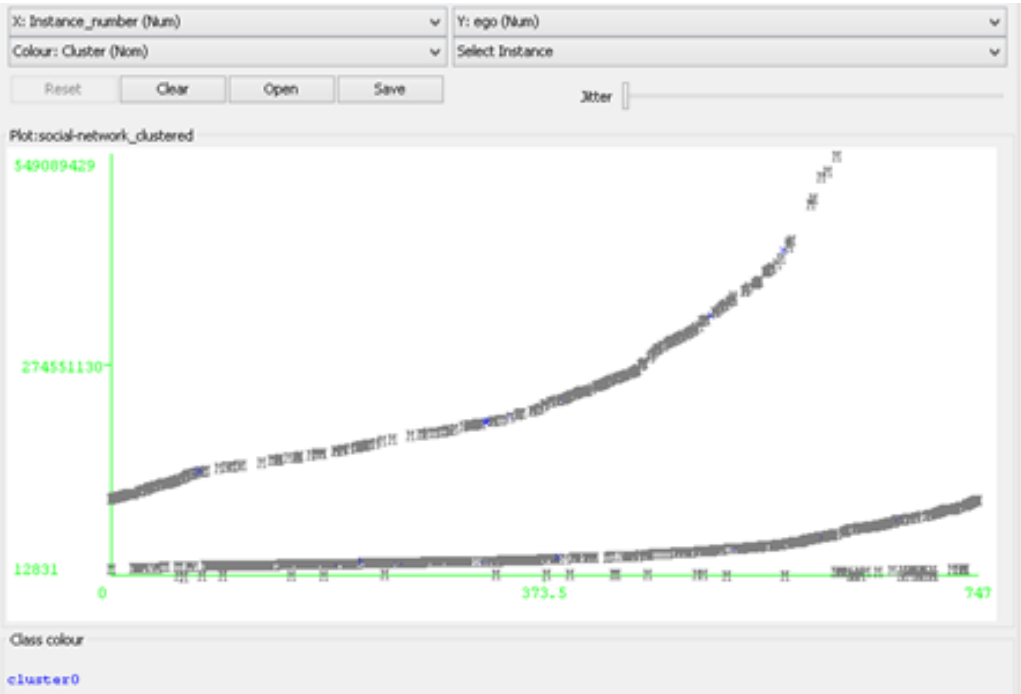
Obrázek 19: Výstupní graf pro 50.000 atributů - COBWEB



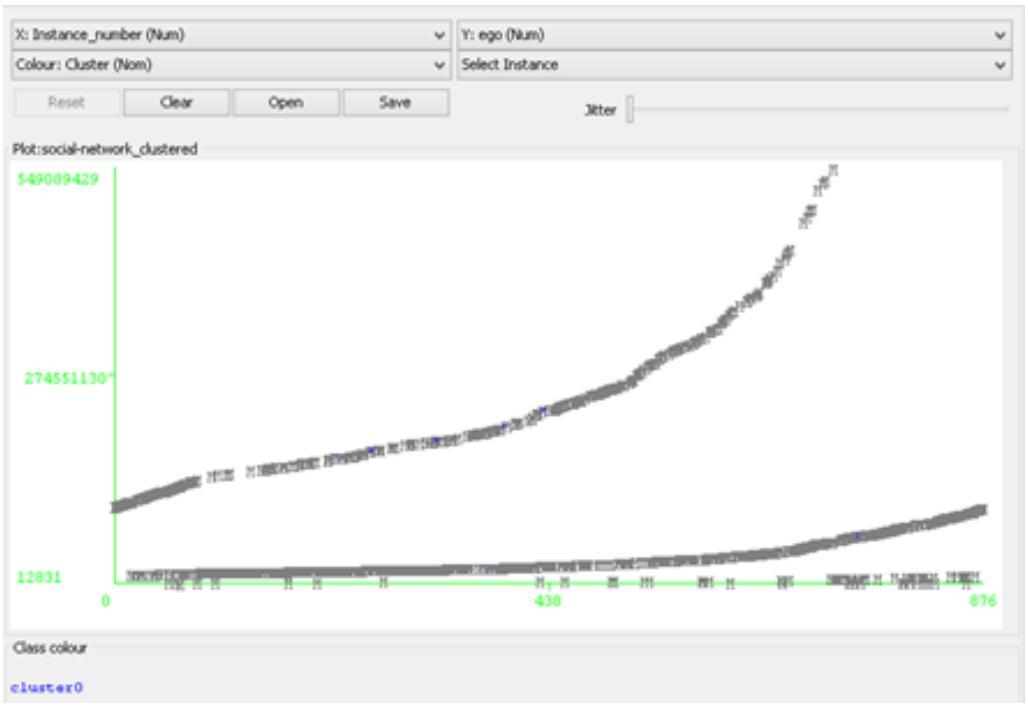
Obrázek 20: Výstupní graf pro 1.000 atributů - DBSCAN



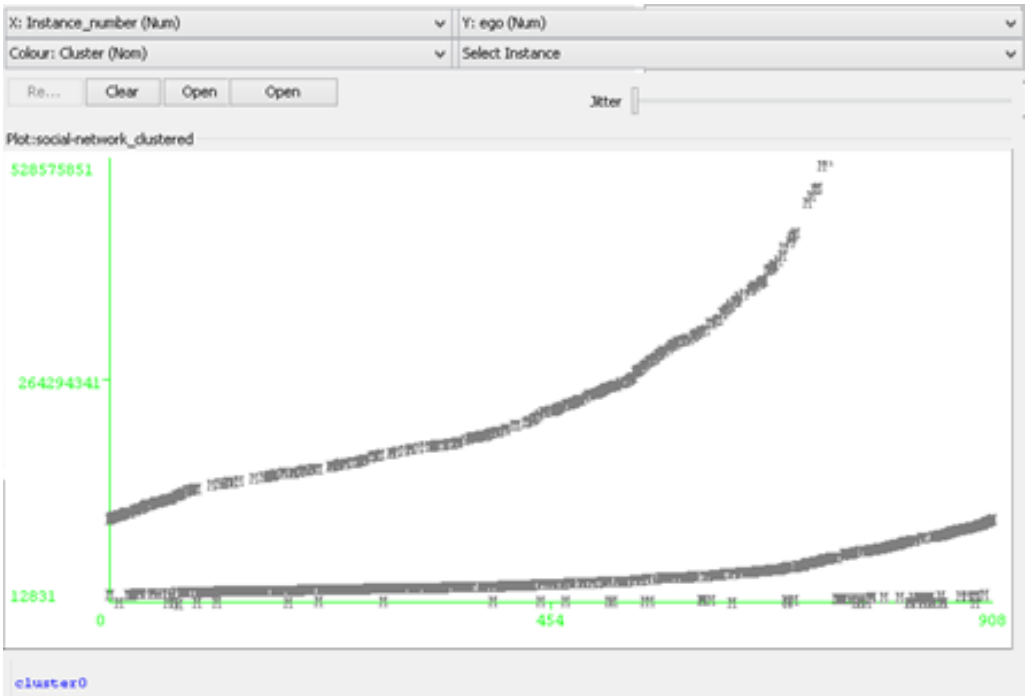
Obrázek 21: Výstupní graf pro 5.000 atributů - DBSCAN



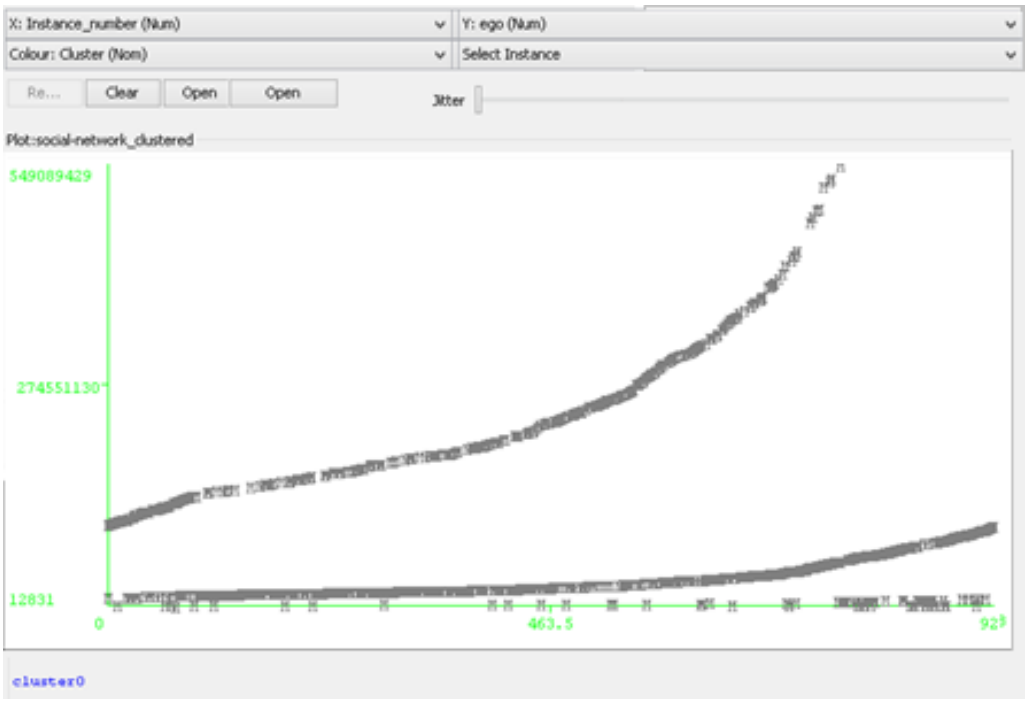
Obrázek 22: Výstupní graf pro 10.000 atributů - DBSCAN



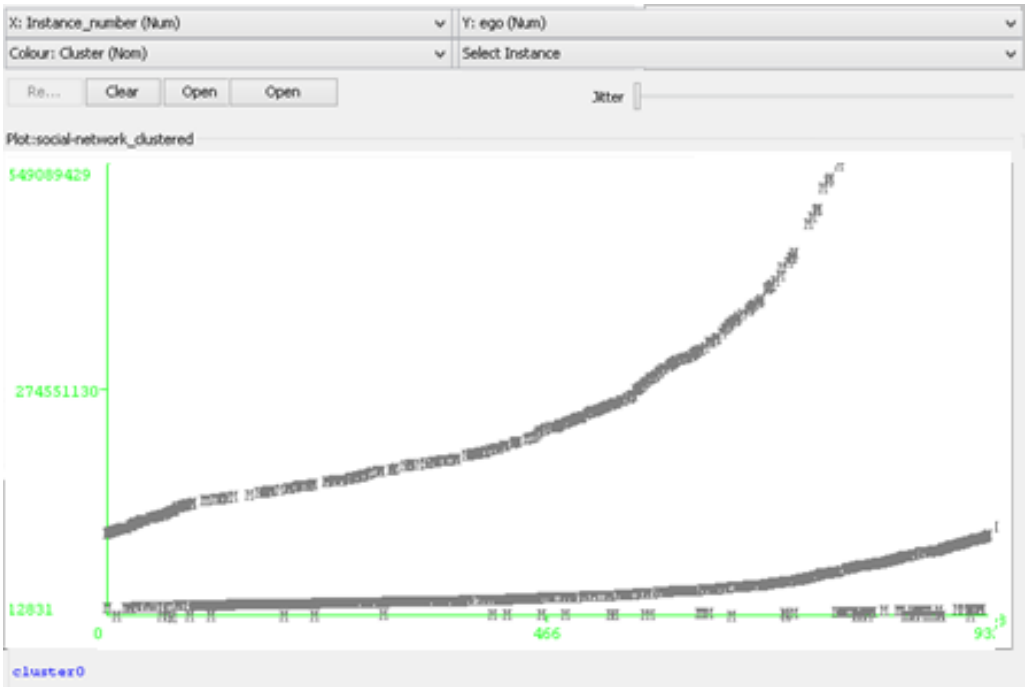
Obrázek 23: Výstupní graf pro 20.000 atributů - DBSCAN



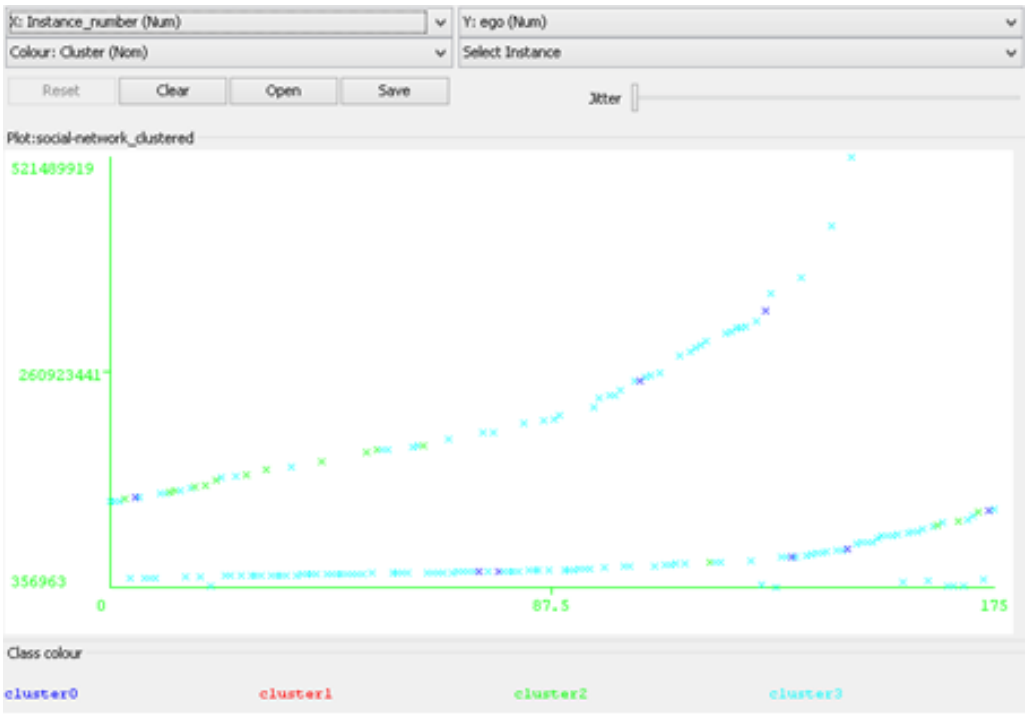
Obrázek 24: Výstupní graf pro 30.000 atributů - DBSCAN



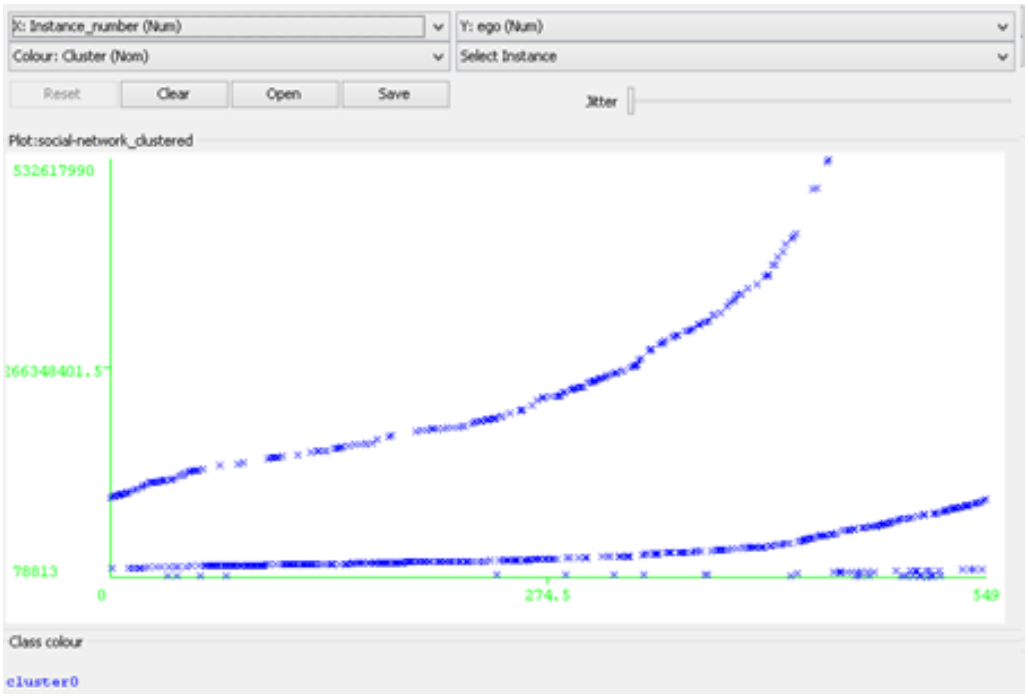
Obrázek 25: Výstupní graf pro 40.000 atributů - DBSCAN



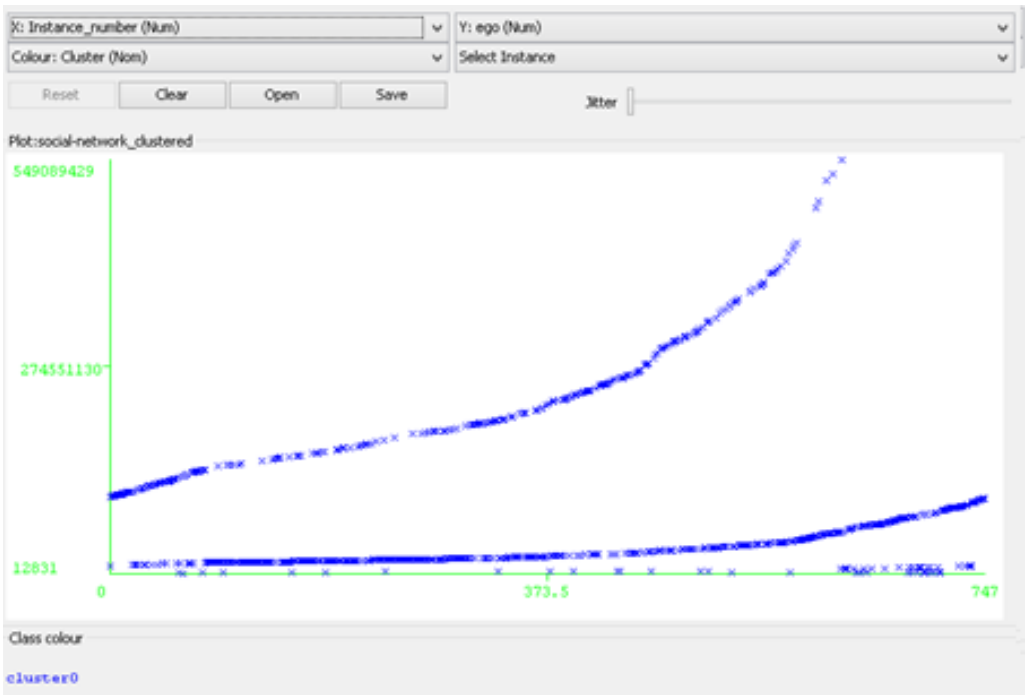
Obrázek 26: Výstupní graf pro 50.000 atributů - DBSCAN



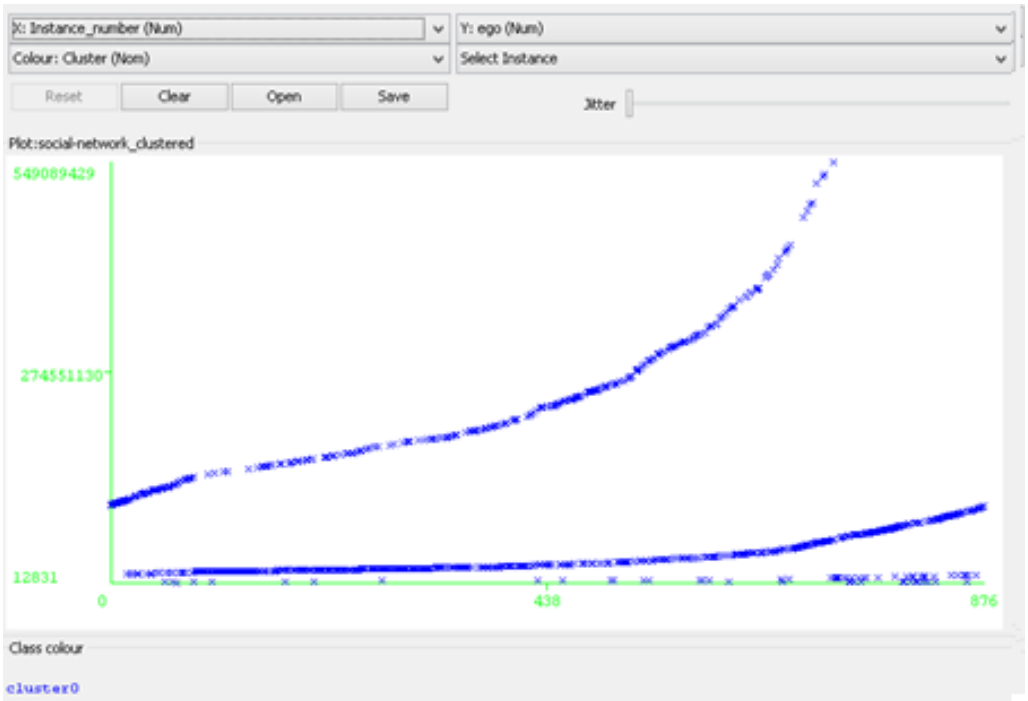
Obrázek 27: Výstupní graf pro 1.000 atributů - EM



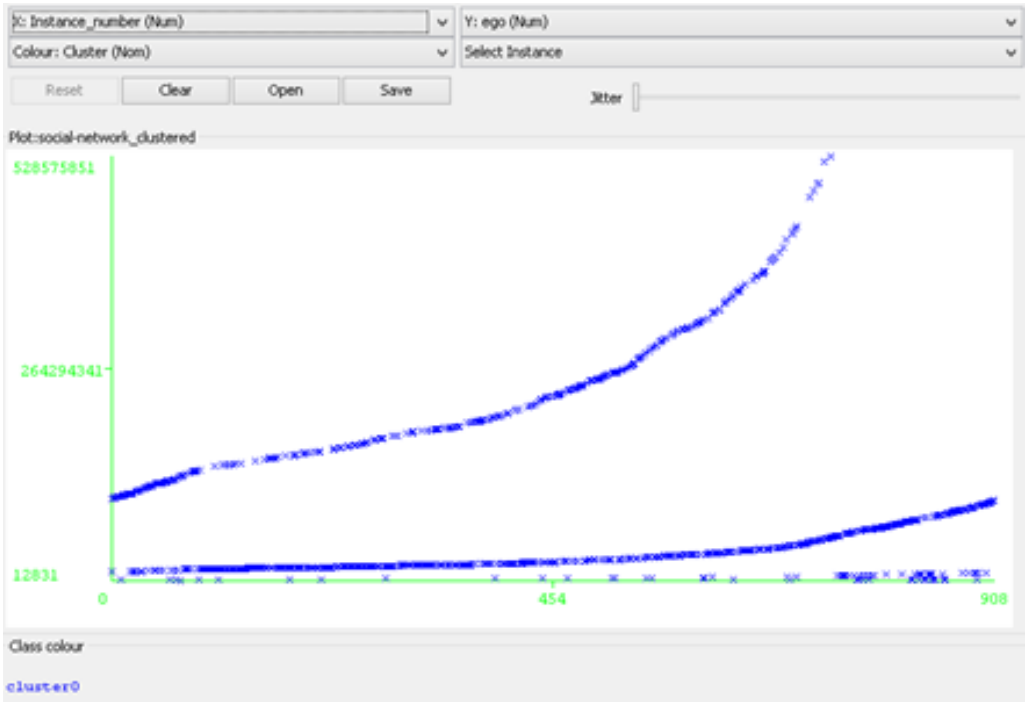
Obrázek 28: Výstupní graf pro 5.000 atributů - EM



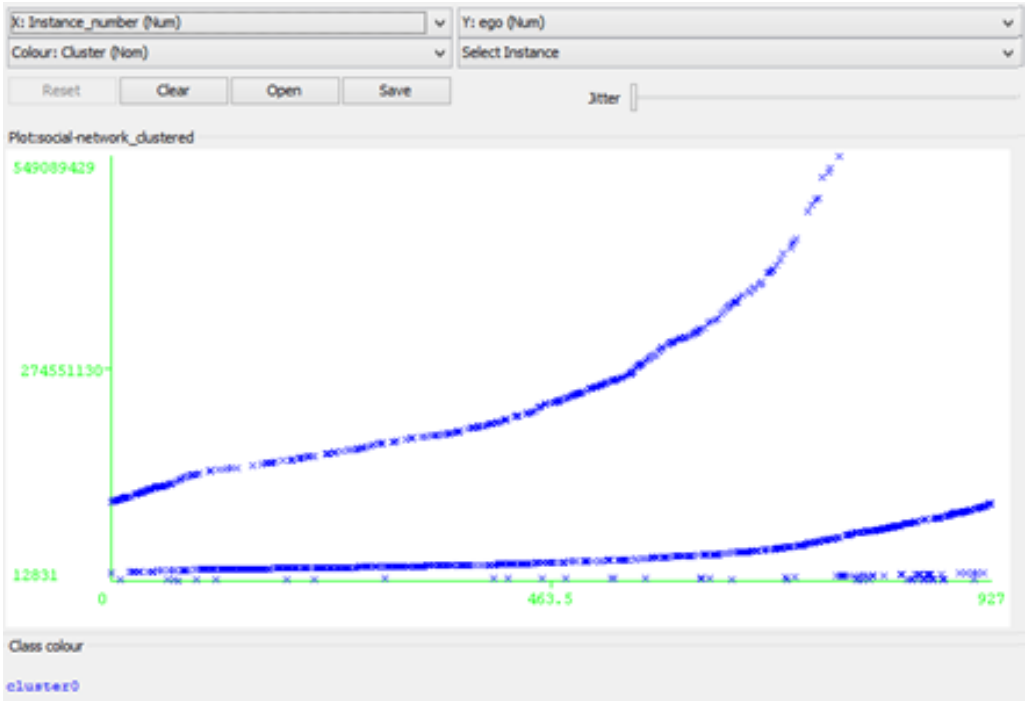
Obrázek 29: Výstupní graf pro 10.000 atributů - EM



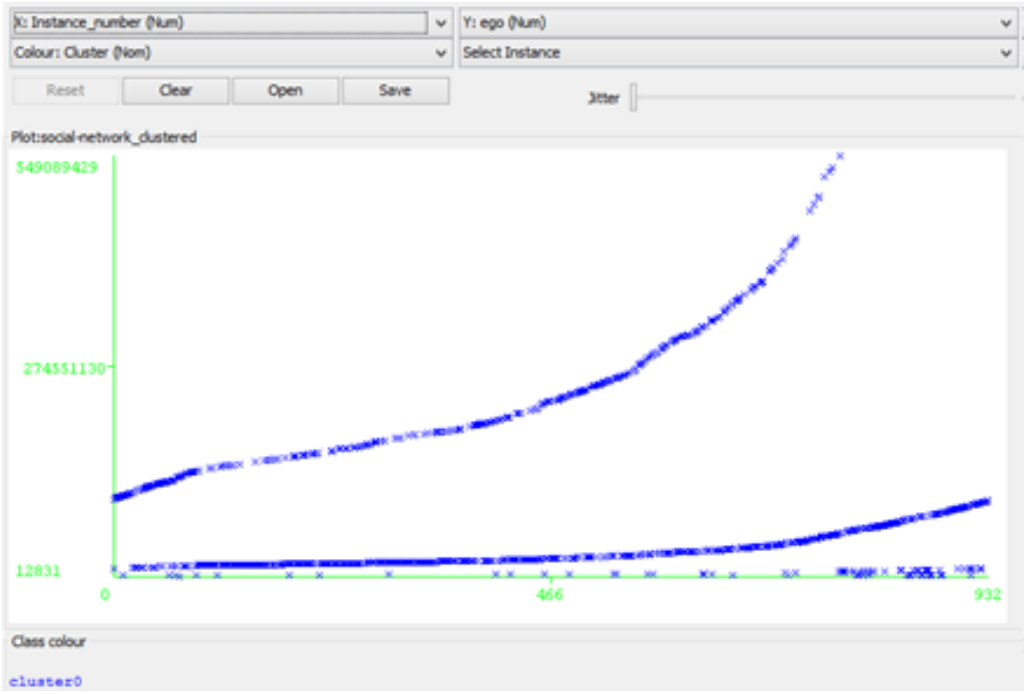
Obrázek 30: Výstupní graf pro 20.000 atributů - EM



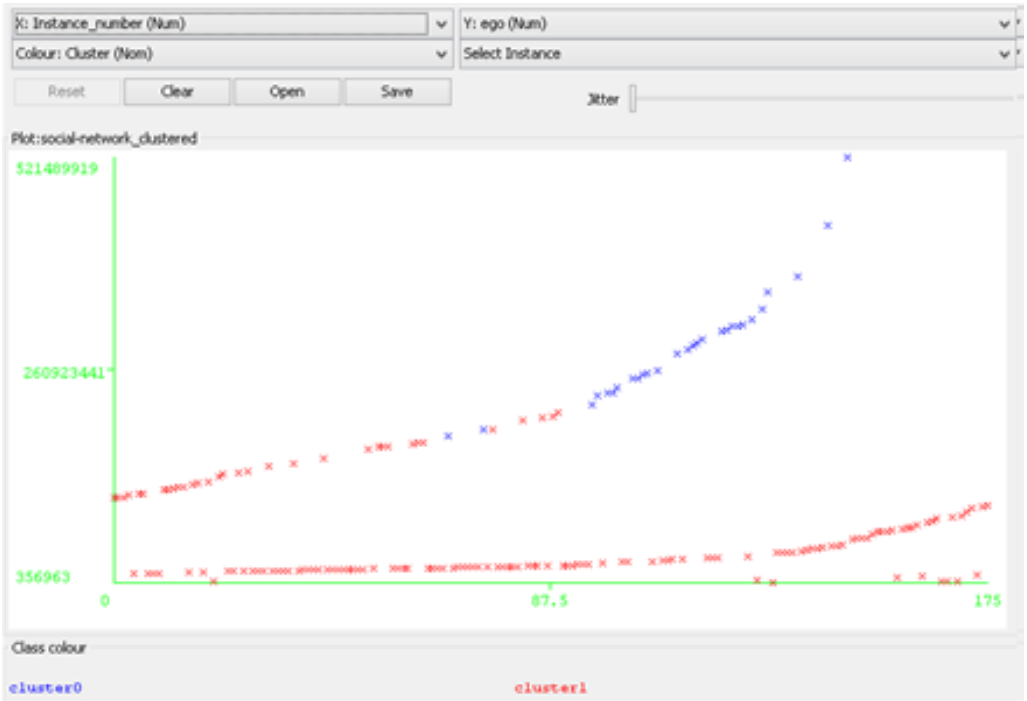
Obrázek 31: Výstupní graf pro 30.000 atributů - EM



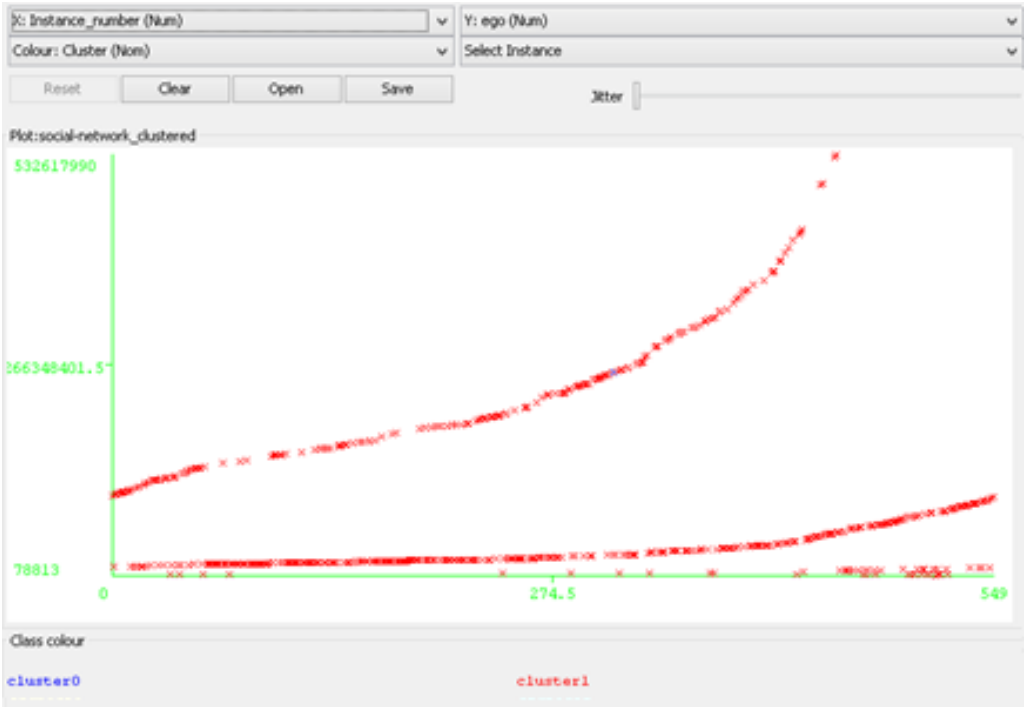
Obrázek 32: Výstupní graf pro 40.000 atributů - EM



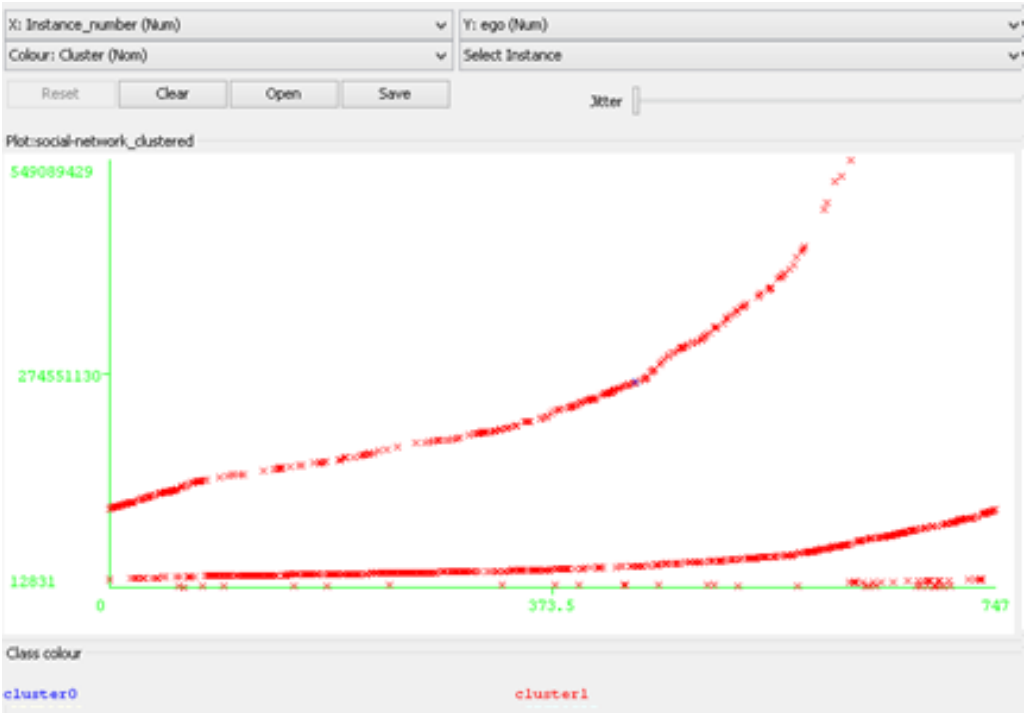
Obrázek 33: Výstupní graf pro 50.000 atributů - EM



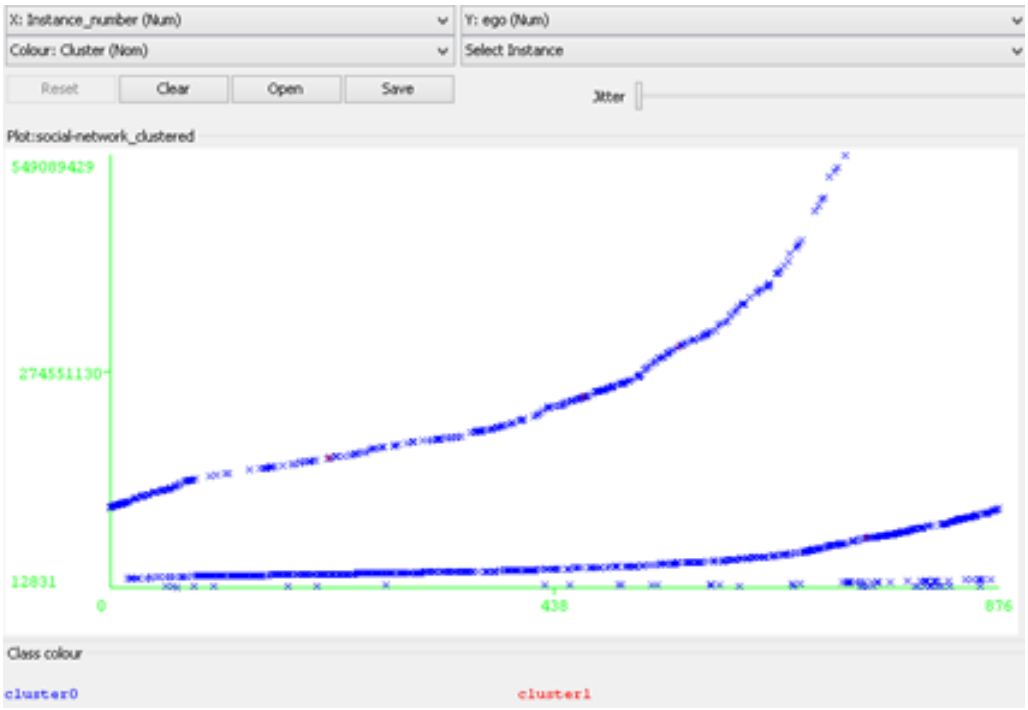
Obrázek 34: Výstupní graf pro 1.000 atributů - K-MEANS



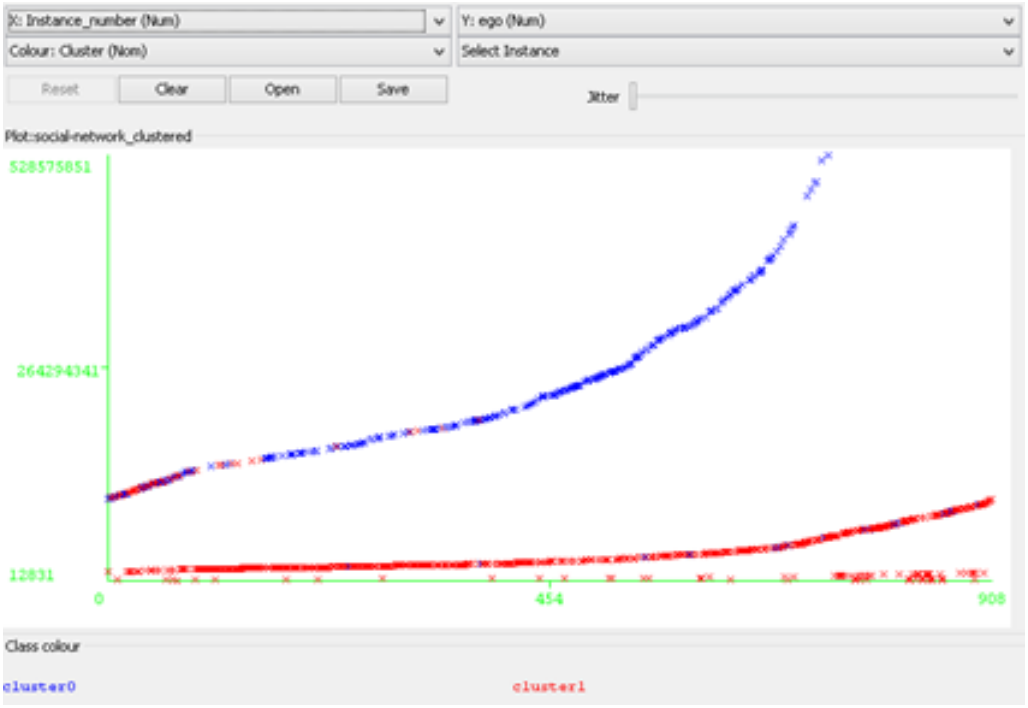
Obrázek 35: Výstupní graf pro 5.000 atributů - K-MEANS



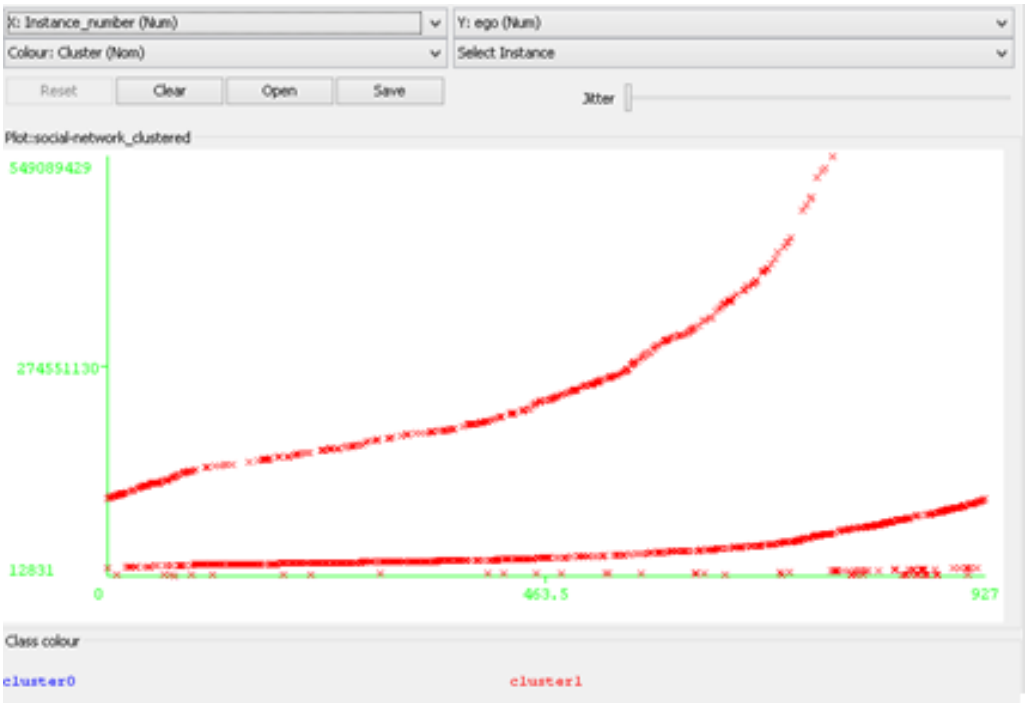
Obrázek 36: Výstupní graf pro 10.000 atributů - K-MEANS



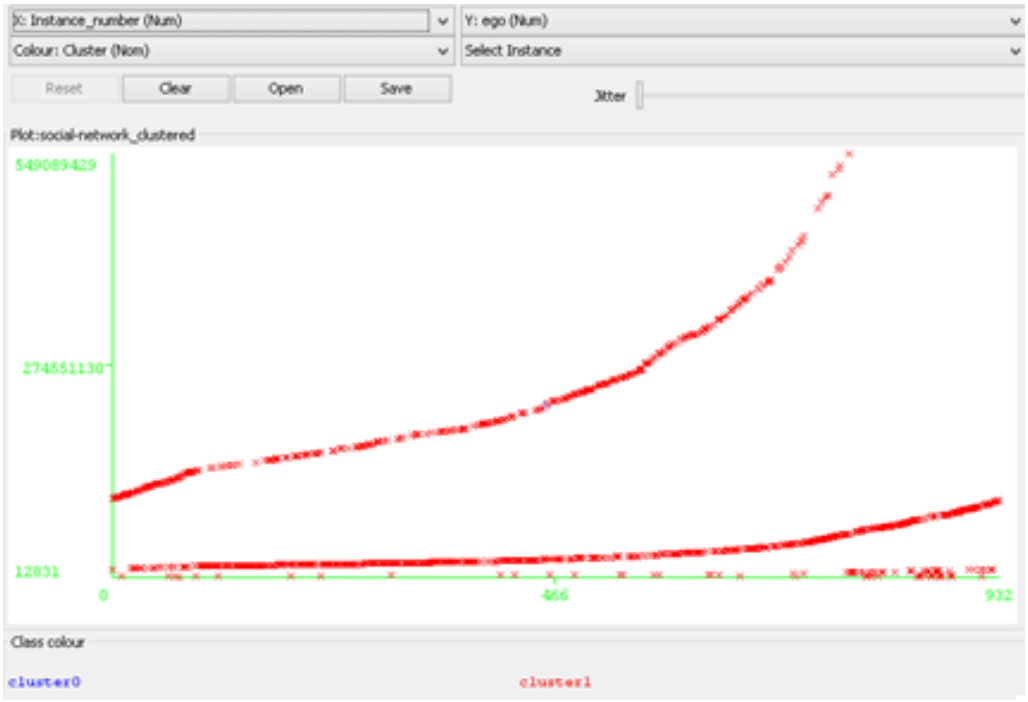
Obrázek 37: Výstupní graf pro 20.000 atributů - K-MEANS



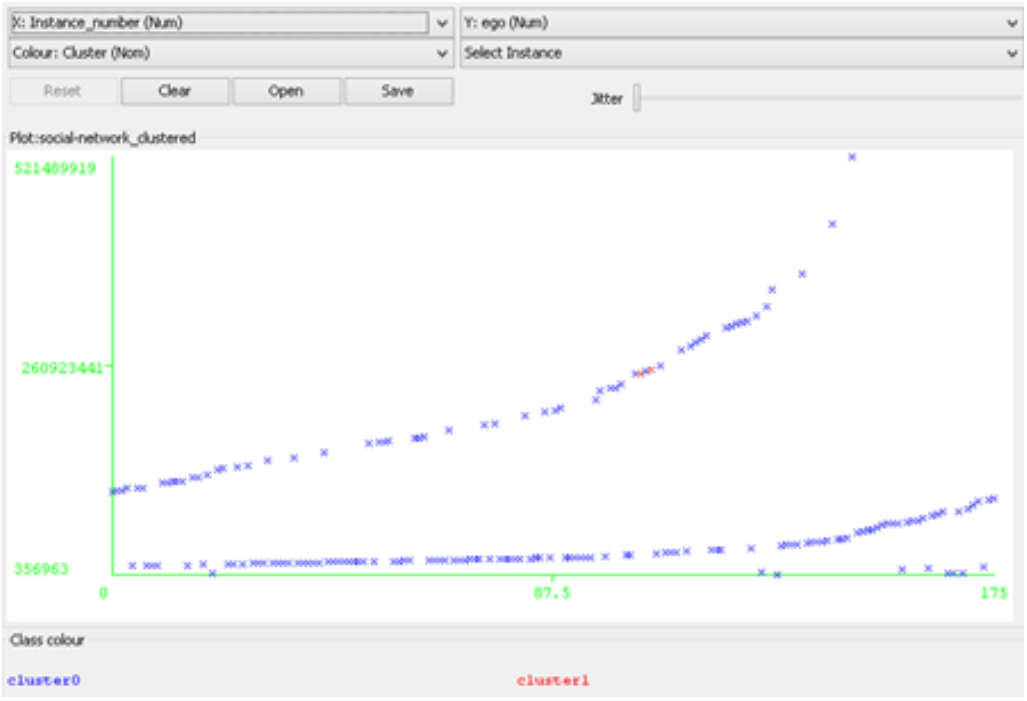
Obrázek 38: Výstupní graf pro 30.000 atributů - K-MEANS



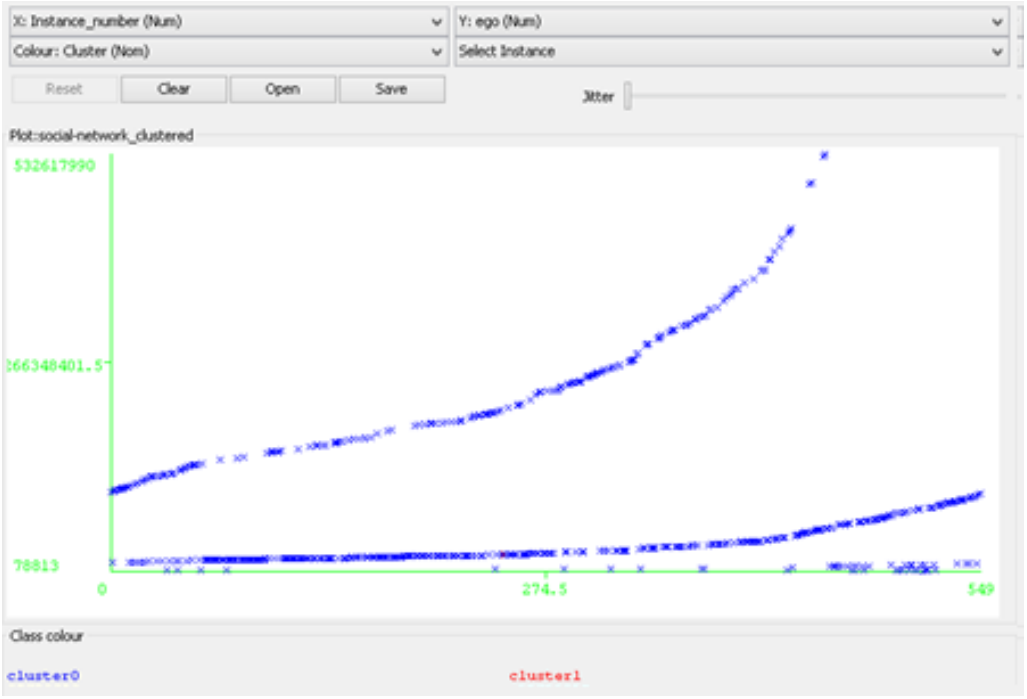
Obrázek 39: Výstupní graf pro 40.000 atributů - K-MEANS



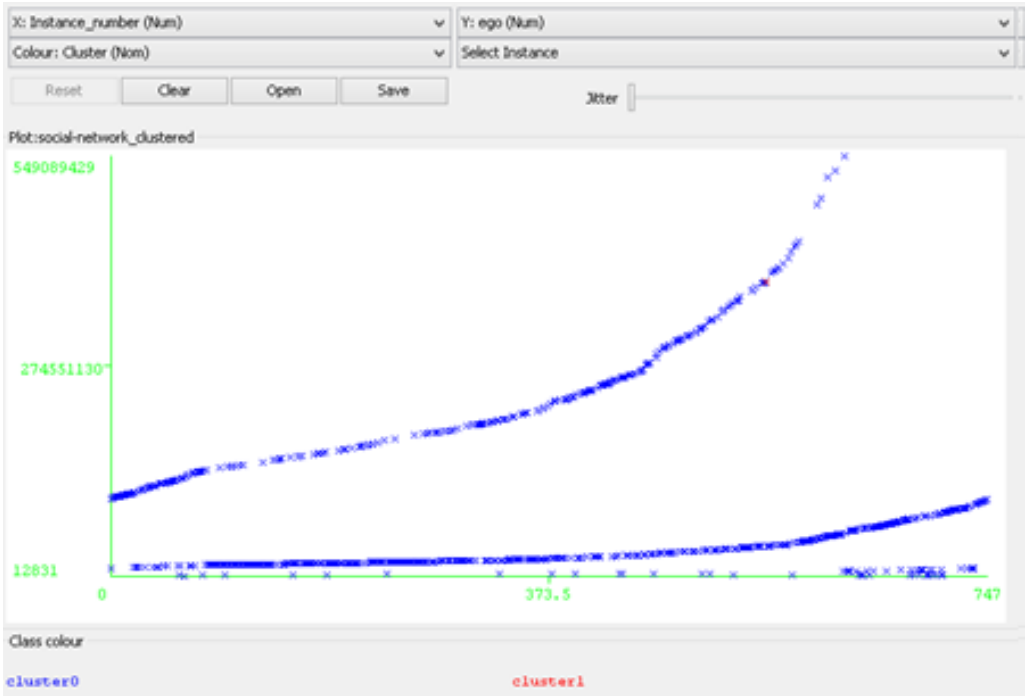
Obrázek 40: Výstupní graf pro 50.000 atributů - K-MEANS



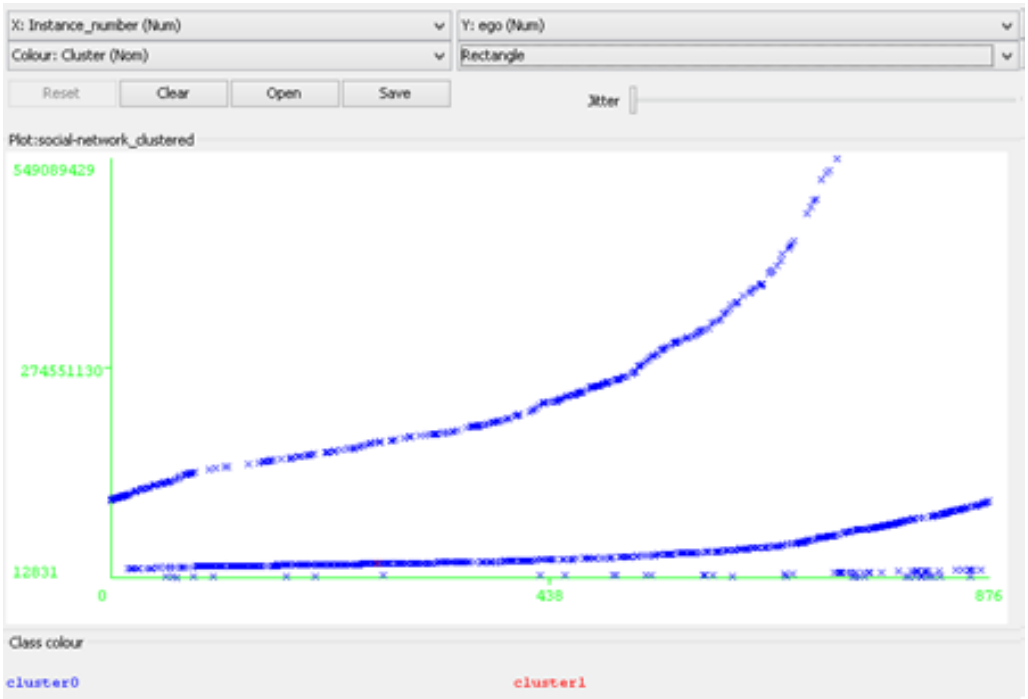
Obrázek 41: Výstupní graf pro 1.000 atributů - FARTHEST FIRST



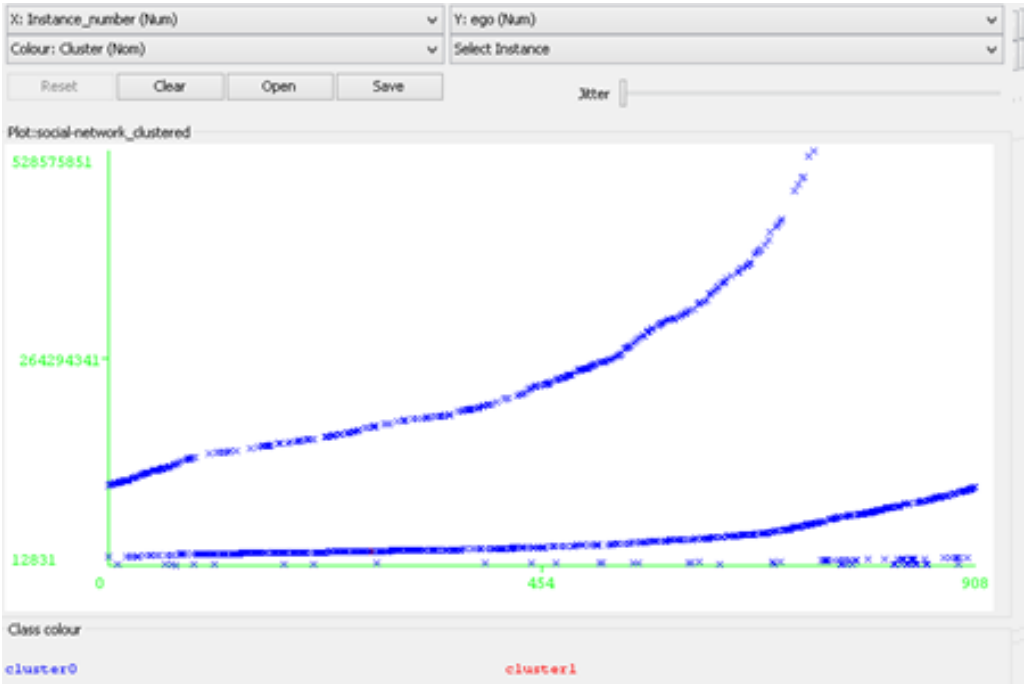
Obrázek 42: Výstupní graf pro 5.000 atributů - FARTHEST FIRST



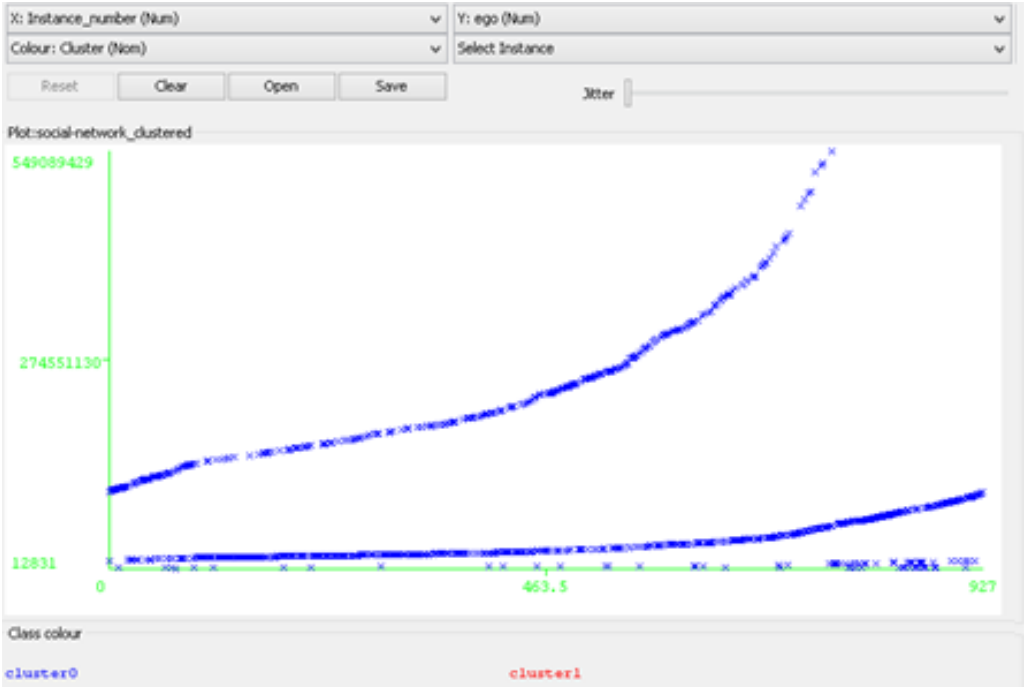
Obrázek 43: Výstupní graf pro 10.000 atributů - FARTHEST FIRST



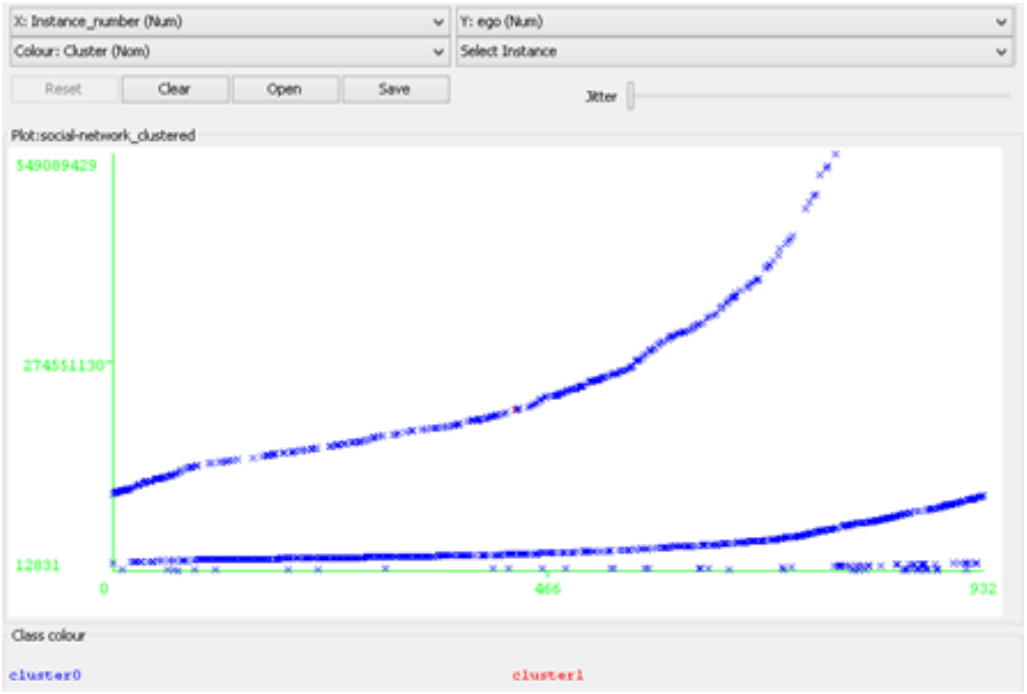
Obrázek 44: Výstupní graf pro 20.000 atributů - FARTHEST FIRST



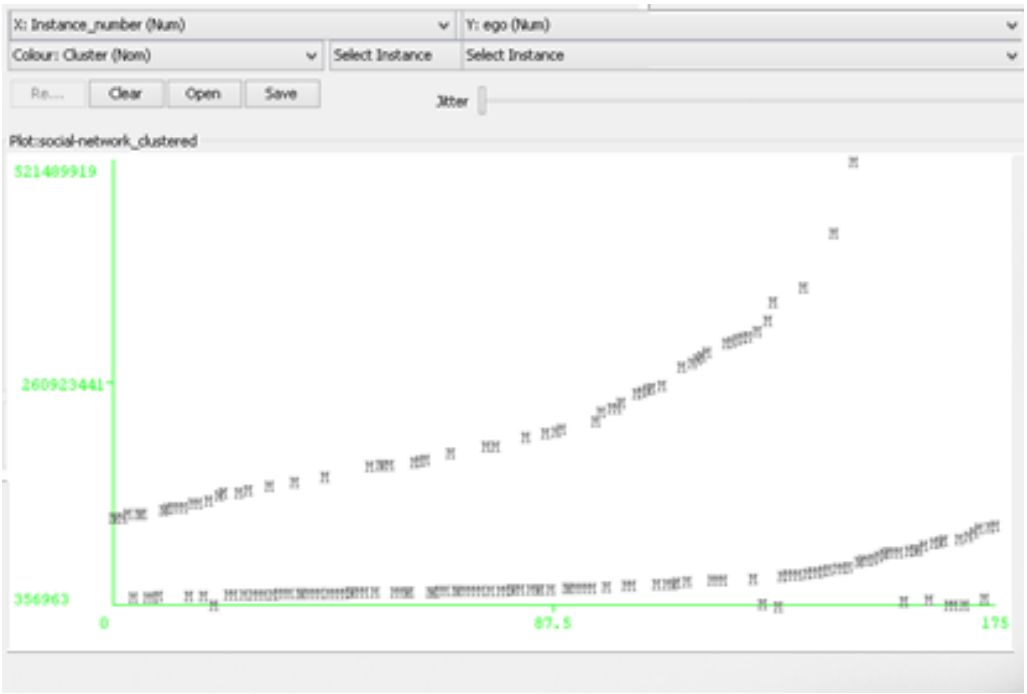
Obrázek 45: Výstupní graf pro 30.000 atributů - FARTHEST FIRST



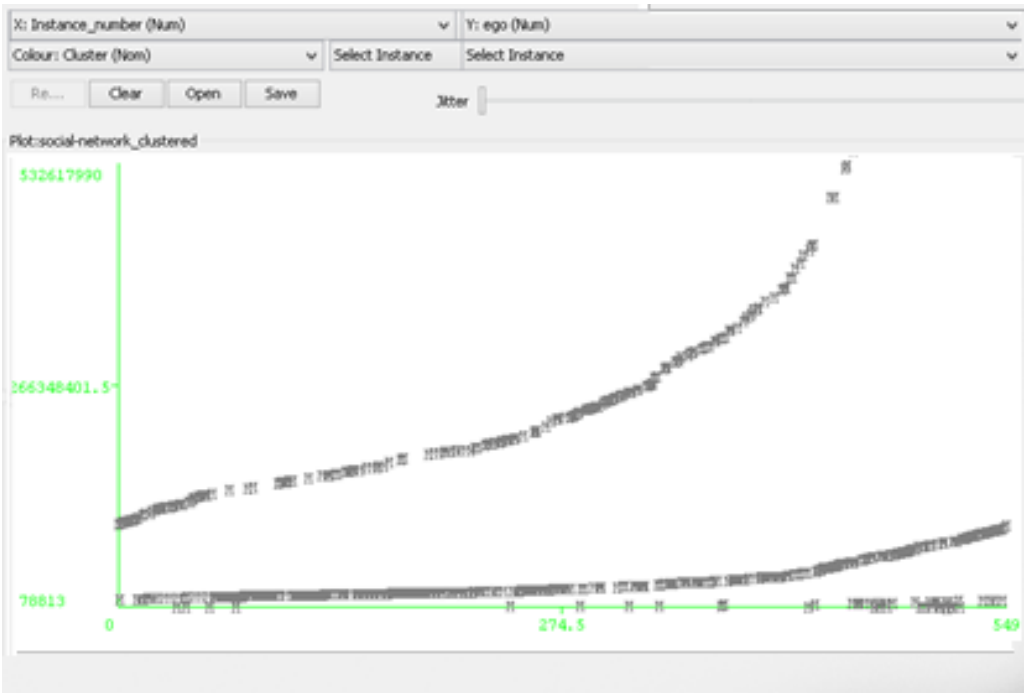
Obrázek 46: Výstupní graf pro 40.000 atributů - FARTHEST FIRST



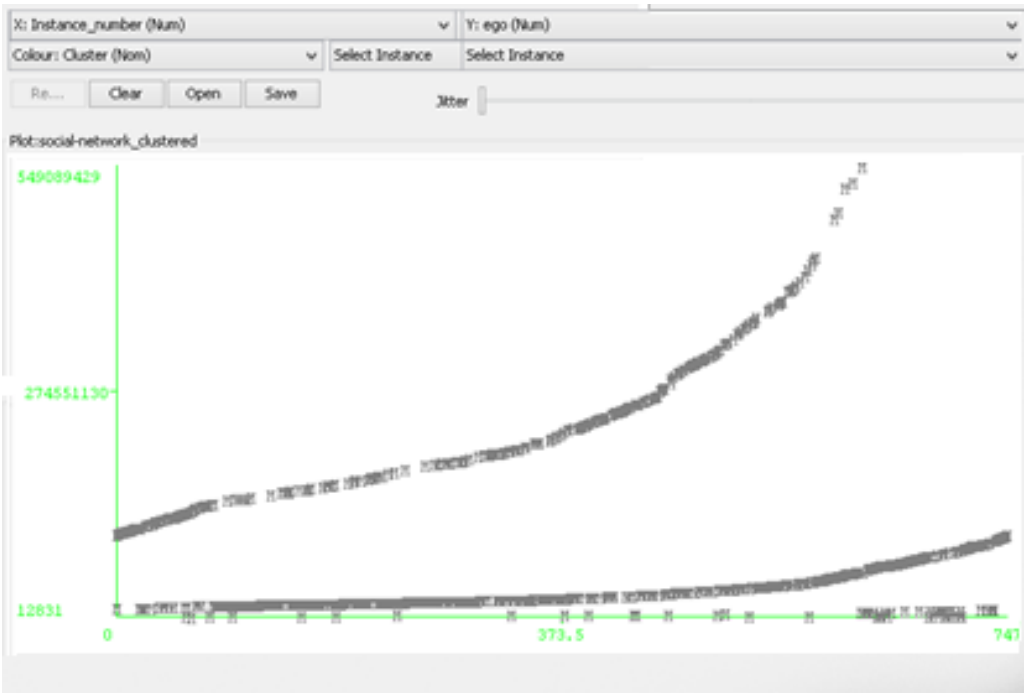
Obrázek 47: Výstupní graf pro 50.000 atributů - FARTHEST FIRST



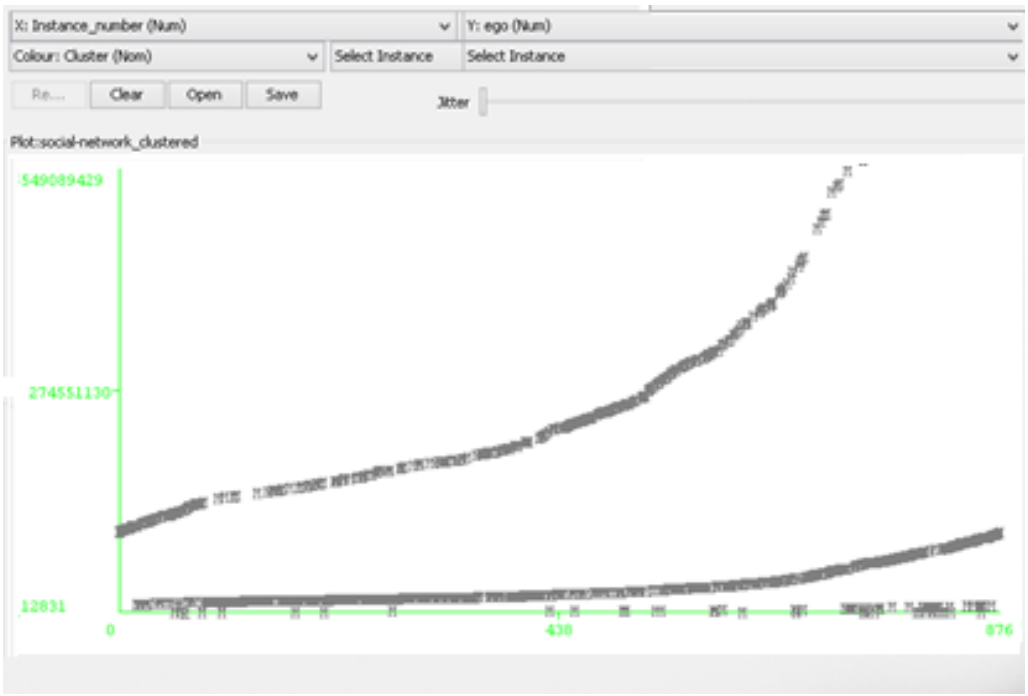
Obrázek 48: Výstupní graf pro 1.000 atributů - OPTICS



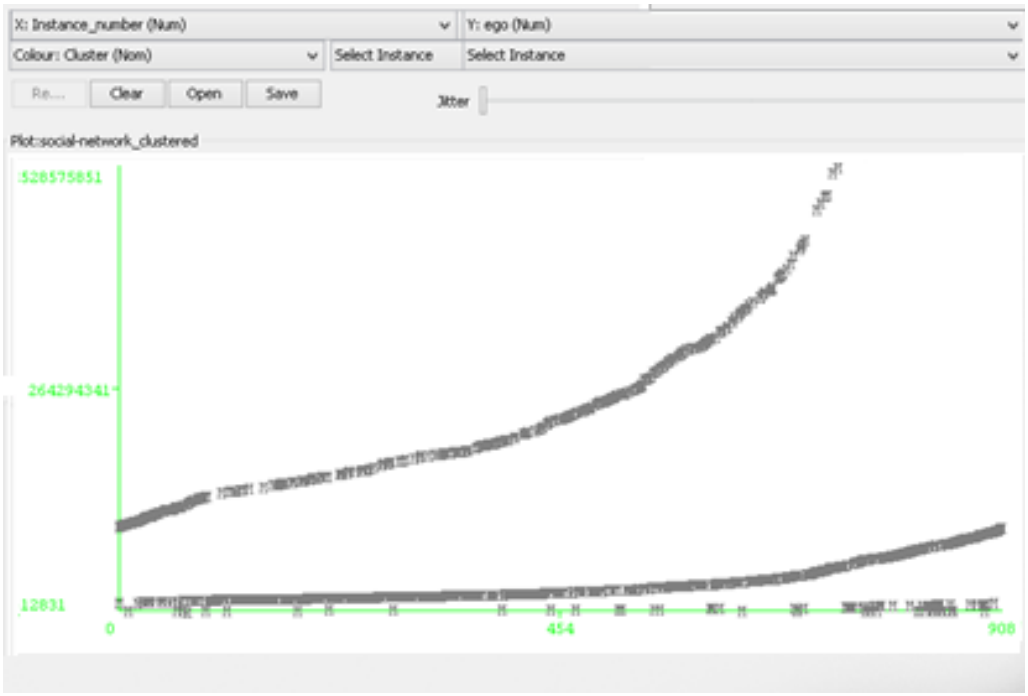
Obrázek 49: Výstupní graf pro 5.000 atributů - OPTICS



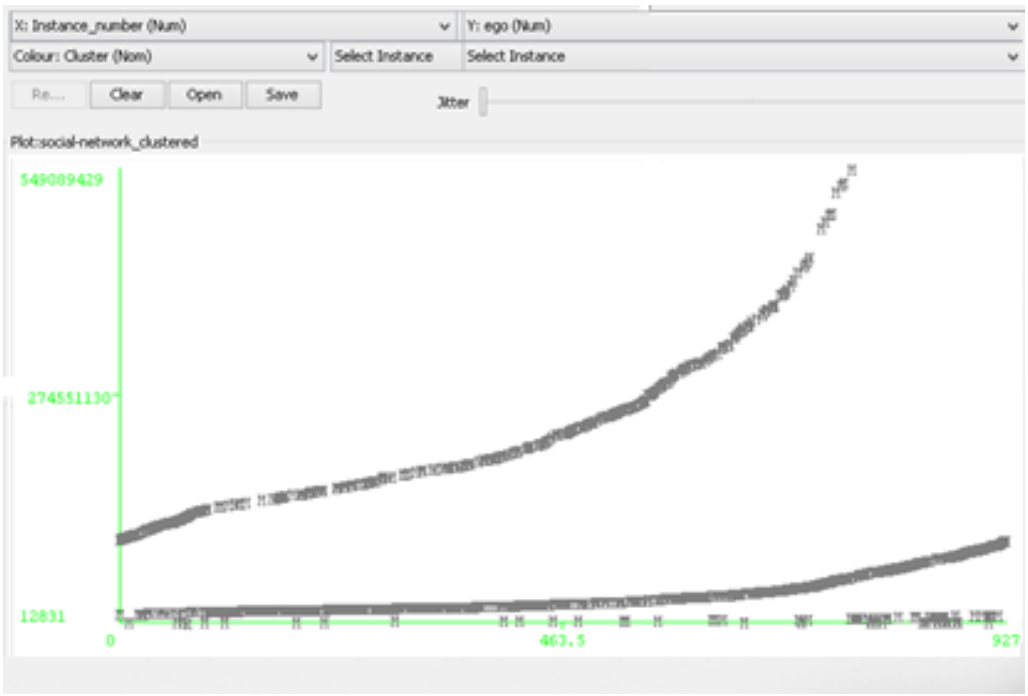
Obrázek 50: Výstupní graf pro 10.000 atributů - OPTICS



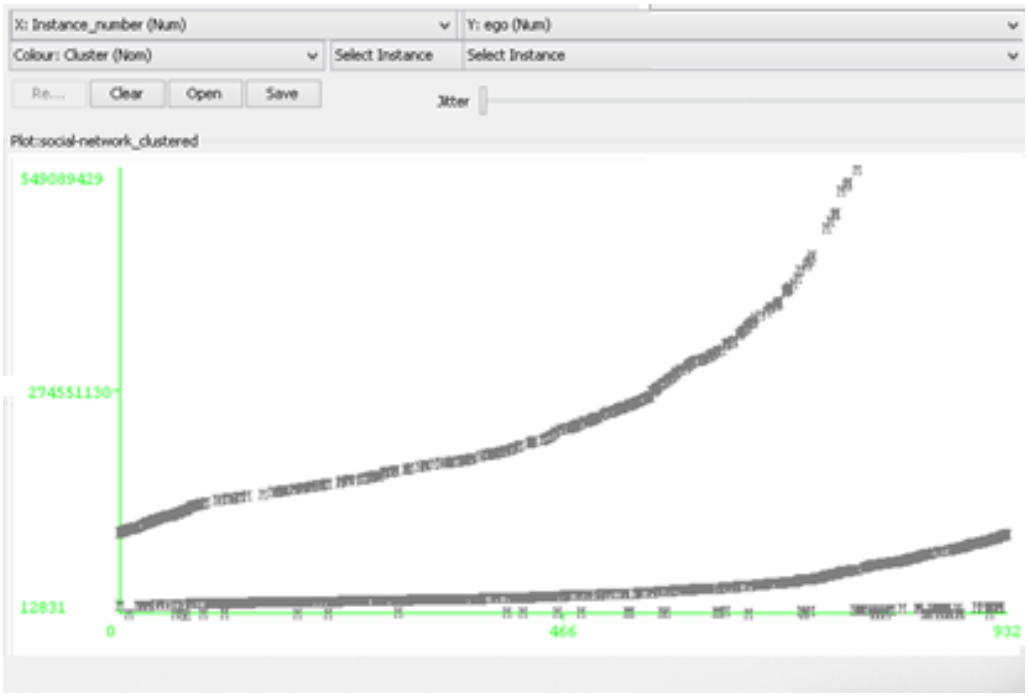
Obrázek 51: Výstupní graf pro 20.000 atributů - OPTICS



Obrázek 52: Výstupní graf pro 30.000 atributů - OPTICS



Obrázek 53: Výstupní graf pro 40.000 atributů - OPTICS



Obrázek 54: Výstupní graf pro 50.000 atributů - OPTICS